TUGAS AKHIR – MN141581

PEMBUATAN PURWARUPA TES MODELUNTUKPENGUJI SISTEM AUTOPILOTPADA*UNMANNED SURFACE VEHICLE* (USV)

Ericza Damaranda S
NRP 4113100068

Dosen Pembimbing
Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.

DEPARTEMEN TEKNIK PERKAPALAN
FAKULTAS TEKNOLOGI KELAUTAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TUGAS AKHIR – MN141581

PEMBUATAN PURWARUPA TES MODELUNTUK MENGUJI SISTEM AUTOPILOTPADA*UNMANNED SURFACE VEHICLE* (USV)

Ericza Damaranda S
NRP 4113100068

Dosen Pembimbing
Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.

DEPARTEMEN TEKNIK PERKAPALAN
FAKULTAS TEKNOLOGI KELAUTAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

FINAL YEAR PROJECT – MN141581

MODEL TEST PROTOTYPE TO EVALUATE AUTOPILOT SYSTEM FORUNMANNED SURFACE VEHICLE *(*USV)

Ericza Damaranda S
NRP 4113100068

Dosen Pembimbing
Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.

DEPARTMENT OF NAVAL ARCHITECTURE
FACULTY OF MARINE TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2017

LEMBAR PENGESAHAN

PEMBUATAN PURWARUPA TES MODELUNTUK MENGUJI SISTEM AUTOPILOTPADA*UNMANNED SURFACE VEHICLE* (USV)

TUGAS AKHIR

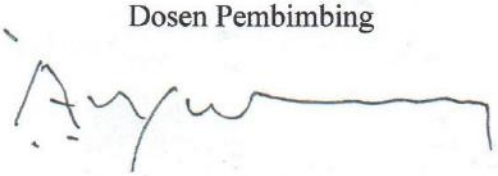
Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Keahlian Rekayasa Perkapalan – Desain Kapal
Program SarjanaDepartemen Teknik Perkapalan
Fakultas Teknologi Kelautan
Institut Teknologi Sepuluh Nopember

Oleh:

ERICZA DAMARANDA S
NRP 4113100068

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dosen Pembimbing



Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.
NIP 19640210 198903 1 001

Mengetahui,
Kepala Departemen Teknik Perkapalan

Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.
NIP 19640210 198903 1 001

SURABAYA, 10 JULI 2017

LEMBAR REVISI

PEMBUATAN PURWARUPA TES MODEL UNTUK MENGUJI SISTEM AUTOPILOT PADA *UNMANNED SURFACE VEHICLE* (USV)

TUGAS AKHIR
Telah direvisi sesuai dengan hasil Ujian Tugas Akhir
Tanggal 7 Juli 2017

Bidang Keahlian Rekayasa Perkapalan – Desain Kapal
Program Sarjana Departemen Teknik Perkapalan
Fakultas Teknologi Kelautan
Institut Teknologi Sepuluh Nopember

Oleh:

ERICZA DAMARANDA S
NRP 4113100068

Disetujui oleh Tim Penguji Ujian Tugas Akhir:

1. Wing Hendroprasetyo A.P., S.T., M.Eng. .....
2. Hasanudin, S.T., M.T. .....
3. Ahmad Nasirudin, S.T., M.Eng. .....

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Ir. Wasis Dwi Aryawan, M.Sc., Ph.D. .....

SURABAYA, 10 JULI 2017

Dipersembahkan untuk keluarga, almamater, dan bangsa

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena atas karunianya Tugas Akhir "**Pembuatan Purwarupa Tes Model Untuk Menguji Sistem Autopilot Pada *Unmanned Surface Vehiche* (USV)**" dapat diselesaikan dengan baik.

Pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang membantu penyelesaian Tugas Akhir ini, yaitu:

1. Bapak Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.selaku Dosen Pembimbing dan Kepala Departemen Teknik Perkapalan ITS;atas bimbingan dan motivasinya selama pengerjaan dan penyusunan Tugas Akhir ini;
2. Bapak Ir. Achmad Zubaydi, M.Eng, Ph.D selaku Dosen Wali selama menjalani masa perkuliahan di Departemen Teknik Perkapalan ITS;
3. Keluarga Penulis, Ibu Tri Damayanti Melani Akas, Bapak Sugito, Mami Yuniy Gemini Arti Akas,Reno Rizal SS, M Davin FS, Abhysa Goldy 'Ibon', Novilia Islamiyati yang selalu memberikan do'a dan dukungan serta motivasi bagi Penulis;
4. Fajar Ramadhan dan Dwiko Hardianto, rekan satu tim Tugas Akhir yang selalu membantu dan menemani selama masa pengerjaan Tugas Akhir. Bambang 'Beng' Kristanto, Titin, Astiti, Indra, dan Armanda yang senantiasa menemani penulis selama masa perkuliahan;
5. Edo dan Andre selaku teman-teman seperjuangan bimbingan Tugas Akhir;
6. Segenap keluarga besar P53 Submarine, Maritime Challenge, Barunastra ITS, dan Robotika ITS selama masa perkuliahan;
7. Dan semua pihak yang telah membantu menyelesaikan Tugas Akhir ini, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan, sehingga kritik dan saran yang bersifat membangun sangat diharapkan. Akhir kata semoga laporan ini dapat bermanfaat bagi banyak pihak.

Surabaya, 10 Juli 2017

Ericza Damaranda S

PEMBUATAN PURWARUPA TES MODELUNTUK MENGUJI SISTEM AUTOPILOTPADA*UNMANNED SURFACE VEHICLE* (USV)

Nama Mahasiswa	:	Ericza Damaranda S
NRP	:	4113100068
Departemen / Fakultas	:	Teknik Perkapalan / Teknologi Kelautan
Dosen Pembimbing	:	Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.

ABSTRAK

Di era teknologi dan komunikasi yang semakin berkembang ini, tugas manusia semakin terbantu dengan hadirnya teknologi berupa robot. Teknologi ini sudah mulai masuk pada dunia transportasi yang salah satunya adalah *Unmanned Surface Vehicle*(USV). Pada saat ini, masalah yang dihadapi oleh TNI AL sebagai institusi yang berwenang dalam pertahanan laut adalah terbatasnya jumlah kapal patroli, jumlah personil, dan dana yang dimiliki. Maka, dengan ini USV diharapkan dapat menggantikan tugas manusia dalam melakukan tugasnya yang berarti dapat mengurangi jumlah personil sehingga kebutuhan dana dapat dialihkan untuk menambah jumlah armada kapal patrol. Penelitian ini bertujuan untuk membuat purwarupa tes model untuk menguji sistem autopilot pada *Unmanned Surface Vehicle*(USV). Platform yang dibuat adalah kapal Katamaran dengan panjang 98.6 cm. Lalu dilanjutkan dengan pembuatan sistem propulsi, sistem autopilot, dan sistem kamera*monitoring*. Sistem kendali autopilot dari USV ini ada dua mode, yaitu kendali dalam mode manual dan kendali dalam mode autopilot itu sendiri. Pada tahap kendali mode manual, pilotmenggunakan *Radio Control* untuk mengendalikan USV. Dalam pengujian menggunakan komunikasi *wirelessRadio Control* 2.4 GHz, pilot dapat dengan efektif mengendalikan USV sampai pada jarak 200 m. Ketika USV sudah berada pada titik operasi dan kecepatan yang diinginkan maka *pilot* akan mengaktifkan sistem kendali mode autopilot. Sistem ini meliputi, sistem navigasi GPS untuk menjaga agar USV berada pada koordinat titik *waypoint* yang telah ditentukan oleh pilot dan kompas untuk mengarahkan USV ke tujuan. Lalu didukung oleh sistem kontrol yang mengemudikan USV mencapai tujuannya. Untuk menampilkan data lokasi USV secara *real time* menggunakan *softwareMission Planner* dengan komunikasi *wireless* berupa *telemetry*. Kemudian untuk dapat menampilkan citra gambar video dari kamera*monitoring* yang terdapat pada USV secara *real time,* menggunakan komunikasi *wireless* 5.8G 600MW 48 Channel AV.

Kata Kunci : Purwarupa, *Unmanned Surface Vehicle,* Sistem Propulsi, Sistem Autopilot, dan Sistem Kamera*Monitoring.*

PROTOTYPE MODEL TEST MAKING TO TEST THE AUTOPILOT SYSTEM IN UNMANNED SURFACE VEHICLE *(*USV)

Author	:Ericza Damaranda S
ID No.	:4113100068
Dept. / Faculty	:Naval Architecture / Marine Technology
Supervisors	:Ir. Wasis Dwi Aryawan, M.Sc., Ph.D.

ABSTRACT

In the era of technology and communication that this growing human tasks, helped by the growing presence of technology in the form of robots. This technology has already started to enter the world of transportation, one of which is the Unmanned Surface Vehicle (USV). On the current problems faced by the TNI AL as an authorized institution in the defence of the sea is the limited number of patrol boats, the number of personnel, and funds. Then, with this USV is expected to replace human tasks in doing their job which means it can reduce the number of personnel so it needs the funds be transferred to the fleet of ships increased the number of patrol. This research aims to create a prototype test models to test the autopilot system on Unmanned Surface Vehicle (USV). The platform is made with a long Catamaran ship 98.6 cm. And continued with the manufacture of propulsion systems, autopilot system, and monitoring camera systems. Autopilot control system from USV these two modes, namely controling manual mode and full mode in autopilot itself. On the stage of the full manual mode, pilots use Radio Control for the control of USV. In testing using wireless communication Radio Control 2.4 GHz, pilots can effectively take control of USV to approximately 200 m. When the USV was already on the operating point and the desired speed then the pilot will enable autopilot mode control system. This includes system, GPS navigation system to keep the USV is at coordinates waypoint specified by the pilot and compass to orient USV to destination. To display location data in real time using USV software Mission Planner with wireless communication in the form of telemetry. Then to be able to display the image of the video image from a camera monitoring the USV in real time, *usingwireless* 5.8G 600MW 48 Channel AV comunication.

Keywords: Prototype, Unmanned Surface Vehicle, Propulsion Systems, Autopilot Systems,and Monitoring Camera Systems.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

I.1. Latar Belakang Masalah

Indonesia dikenal sebagai Negara Maritim atau Negara Kepulauan terbesar di dunia dengan total luas negara sebesar 5.193.250 km^2 yang mencakup luas daratan dan lautan. Dimana $^2/_3$ luas Indonesia adalah lautan dengan luasan sebesar 3.257.483 km^2 (Wikipedia). Sehingga banyak pulau – pulau di Indonesia yang terpisah karena lautan. Lautan Indonesia pun memiliki batas sesuai hukum laut Internasional, yaitu dengan menggunakan teritorial laut sepanjang 12 mil laut serta Zona Ekonomi Eksklusif (ZEE) sepanjang 200 mil laut yang searah dengan penjuru mata angin. Hal ini harus mendapat perhatian khusus dari pemerintah untuk mengamankan kedaulatan negara. Alat Utama Sistem Senjata atau dikenal dengan ALUTSISTA menjadi kunci utama penjaga keamanan nagara. Sehingga diperlukannya kapal patroli sebagai kapal pengintai yang dapat menjaga kedaulatan wilayah laut Indonesia yang akhir-akhir ini sedang ramai diperbincangkan, mulai dari penculikan Indonesia di wilayah perbatasan, penyelundupan barang ilegal, hingga kasus ilegal fishing oleh negara lain.

Sementara, masalah yang dihadapi oleh TNI AL sebagai institusi yang berwenang dalam pertahanan laut adalah terbatasnya jumlah kapal patroli, jumlah personil, dan dana yang dimiliki. Di era teknologi dan komunikasi yang semakin berkembang ini, tugas manusia semakin terbantu dengan hadirnya teknologi berupa robot. Teknologi ini sudah mulai masuk pada dunia transportasi yang salah satunya adalah kapal tanpa awak. Dengan adanya teknologi robot kapal tanpa awak atau yang biasa disebut USV (*Unmanned Surface Vehicle),* maka kapal tersebut diharapkan dapat menggantikan tugas manusia dalam melakukan tugasnya yang berarti dapat mengurangi jumlah personil sehingga kebutuhan dana dapat dialihkan untuk menambah jumlah armada kapal patrol USV.

Oleh karena itu, pada Tugas Akhir ini akan dikembangkan purwarupa tes model kapal tanpa awak dalam skala kecil, juga akan dilakukan pengujian sistem propulsi, sistem autopilot, dan sistem kamera pada *Unmanned Surface Vehicle* (USV). Pengujian ini meliputi 2 metode pengendalian, yaitu pengendalian kapal secara manual menggunakan *remote control* dan pengendalian kapal secara autopilotdengan menentukan tujuan pergerakan berdasarkan titik-titik koordinat GPS menggunakan metode *waypoint* yang dipandu oleh kompas. Selain itu, GPS

juga digunakan untuk mengetahui letak dari USV sehingga kapal tersebut dapat dipantau secara *real time* melalui layar monitor. Sedangkan untuk mengetahui arah mata angin digunakan sensor kompas. Pengiriman data GPS dan kompas ke layar monitor menggunakan mikrokontroler yang ditransfer dengan *transmitter* secara *wireless*. Kemudian untuk dapat melakukan pengawasan saat pengintaian, USV menggunakan kamera yang akan mengirimkan citra gambar video secara *real time* ke *Ground Control Station* (GSC). Dengan dibuatnya alat ini diharapkan memberi kemudahan dalam pemantauan area perbatasan dengan menggunakan kendaraan air tanpa awak. Harapannya, ke depannya purwarupa ini akan dapat dibuat dalam skala aslinya.

I.2. Perumusan Masalah

Berdasarkan latar belakang di atas, beberapa permasalahan yang akan diselesaikan adalah sebagai berikut:

1. Bagaimana pembuatanpurwarupa tes model *Unmanned Surface Vehicle* (USV)?
2. Bagaimana desain dan pembuatan sistem propulsi pada purwarupa tes model *Unmanned Surface Vehicle* (USV)?
3. Bagaimana desain dan pembuatan sistem autopilot pada purwarupa tes model *Unmanned Surface Vehicle* (USV)?
4. Bagaimana desain dan pembuatan sistem kamera *monitoring* pada purwarupa tes model *Unmanned Surface Vehicle* (USV)?
5. Bagaimana hasil pengujian sistem propulsi pada purwarupa tes model*Unmanned Surface Vehicle* (USV)?
6. Bagaimana hasil pengujian sistem autopilot pada purwarupa tes model *Unmanned Surface Vehicle* (USV)?
7. Bagaimana hasil pengujian sistem kamera *monitoring* pada purwarupa tes model *Unmanned Surface Vehicle* (USV)?

I.3. Tujuan

Tujuan dari penulisan Tugas Akhir ini adalah sebagai berikut:

1. Membuat platform purwarupa tes model *Unmanned Surface Vehicle* (USV).
2. Mendesain dan membuat sistem propulsi pada purwarupa test model *Unmanned Surface Vehicle* (USV).
3. Mendesain dan membuat sistem autopilot pada purwarupa test model *Unmanned Surface Vehicle* (USV).

4. Mendesain dan membuat sistem kamera *monitoring* pada purwarupa test model *Unmanned Surface Vehicle* (USV).
5. Menguji pengendalian manual purwarupa test model *Unmanned Surface Vehicle* (USV).
6. Menguji pengendalian autopilot purwarupa test model *Unmanned Surface Vehicle* (USV).
7. Menampilkan lokasi *Unmanned Surface Vehicle* (USV) pada layar monitor secara *real time.*
8. Menampilkan citra gambar video dari kamera*monitoring* secara *real time* dari *Unmanned Surface Vehicle* (USV) ke *Ground Control Station* (GSC).

I.4. Batasan Masalah

Batasan masalah dalam Tugas Akhir ini adalah sebagai berikut:

1. Tanpa melakukan perhitungan perancangan konstruksi kapal pada model purwarupa *Unmanned Surface Vehicle* (USV).
2. Sensor yang dibahas terbatas hanya GPS dan kompas.
3. Akses dan kalibrasiprogram hanya terbatas pada GPS dan kompas dengan bahasa algoritma yang sudah ada di *library*.
4. Pengujian purwarupa dilakukan di danau 8 ITS, tidak ada ombak besar dan penghalang pada jalur yang akan dilalui.
5. Data yang digunakan adalah data sekunder dari Tugas Akhir Dwiko Hardianto yang berjudul "Pembuatan Konsep Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia" dan Fajar Ramadhan yang berjudul "Pembuatan Detail Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia".

I.5. Manfaat

Dari Tugas Akhir ini, diharapkan dapat diambil manfaat sebagai berikut:

1. Mendukung penelitian dan pengembangan tentang kapal tanpa awak di Indonesia.
2. Mendukung tugas pemerintahan Indonesia dalam menjaga keamanan dan kedaulatan perairan Indonesia khususnya TNI AL,dengan tugas *monitoring* perairan.

I.6. Hipotesis

Purwarupa *Unmanned Surface Vehicles* (USV) ini dimungkinkan dapat dikembangkan lebih lanjut untuk membantu tugas patroli TNI AL dalam melindungi keamanan dan kedaulatan perairan Negara Kesatuan Republik Indonesia (NKRI).

BAB II
STUDI LITERATUR

II.1. Dasar Teori

Pada Bab II ini berisikan tentang dasar teori dan tinjauan pustaka dari topik utama dalam pembuatan Tugas Akhir ini. Dasar teori berisiuraian singkat tentang landasan teori yang mempunyai keterkaitan langsung dan digunakan untuk menyelesaikan permasalahan dalam Tugas Akhir ini.

II.1.1. Robot

Robot adalah seperangkat alat mekanik yang bisa melakukan tugas fisik, baik dengan pengawasan dan kontrol manusia, ataupun menggunakan program yang telah didefinisikan terlebih dulu (kecerdasan buatan). Istilah robot berawal bahasa Ceko "*robota*" yang berarti pekerja atau kuli yang tidak mengenal lelah atau bosan. Robot biasanya digunakan untuk tugas yang berat, berbahaya, pekerjaan yang berulang dan kotor. Biasanya kebanyakan robot industri digunakan dalam bidang produksi. Penggunaan robot lainnya termasuk untuk pembersihan limbah beracun, penjelajahan bawah air dan luar angkasa, pertambangan, pekerjaan "cari dan tolong" (*search and rescue*), dan untuk pencarian tambang. Belakangan ini robot mulai memasuki pasaran konsumen di bidang hiburan, dan alat pembantu rumah tangga, seperti penyedot debu, dan pemotong rumput(www.wikipedia.com)

Robot dibagi menjadi dua jenis sesuai klasifikasinya, yaitu :

1. *Industrial Robot,* yaitu robot yang tidak dapat berpindah posisi dari satu tempat ke tempat lainnya, sehingga robot tersebut hanya dapat menggerakkan beberapa bagian dari tubuhnya dengan fungsi tertentu yang telah dirancang.Contoh : robot manipulator berlengan

Gambar II.1 Robot Manipulator Berlengan
(Sumber : https://www.robots.com/media/1457623303_1.jpg)

2. *Mobile Robot,* yaiturobot yang dapat berpindah posisi dari satu titik ke titik yang lain secara mandiri. Supaya bisa berpindah secara mandiri, robot kapal harus memiliki sistem navigasi. Fungsi sistem navigasi itu sendiri sebagai alat pemandu robot kapal agar dapat berpindah tempat. Contoh : *Unmanned Surface Vehicle*

Gambar II.2*Unmanned Surface Vehicle* (USV)
(Sumber : Damaranda, 2016)

II.1.2. *Unmanned Surface Vehicle*(USV)

Istilah *Unmanned Surface Vehicle (USV)* atau *Autonomous Surface Vehicle (ASV)* dimaksudkan untuk wahana yang dioperasikan pada permukaan air tanpa awak. USV dikendalikan autopilot dengan memberikan perintah- perintah seperti *waypoint*, melalui *Ground Control Station* (GCS). USV dapat mengolah data-data pergerakan dan pengamatan lalu mengirimkannya ke GCS secara *real time* melalui sistem komunikasi nirkabel. USV sebenarnya telah diujicobakan sejak Perang Dunia II, namun belum terlalu dikenal karena kapal-kapal pengintai tanpa awak generasi awal seperti OWL Mk II masih diklasifikasikan

sebagai *Autonomous Underwater Vehicle* (AUV). Dilanjutkan tahun 1944, Kanada mengembangkan konsep torpedo COMOX sebelum penyerangan ke Normandia. Setelah Perang Dunia II penggunaan USV menjadi lebih berkembang, di antaranya untuk mengambil sampel air yang terkena radioaktif setelah pemboman Able dan Baker di Atol Bikini tahun 1946. USV juga sangat berguna dalam bidang pengamatan wilayah, pengintaian, dan patrol karena memiliki kemampuan yang lebih baik dibandingkan menggunakan radar, namun lebih murah dibandingkan memperbanyak kapal komersial maupun pesawat terbang serta lebih fleksibel daripada yang dapat dilakukan oleh kapal komersial.

Beberapa jenis USV saat ini akan menjadi peralatan yang efisien untuk melakukan patroli, seperti patroli patok-patok perbatasan di laut, perairan dangkal dan di sekitar garis pantai. Sistem ini dapat mengisi kekosongan data yang tidak dapat dilakukan oleh kapal patroli biasa. Untuk berpatroli di patok perbatasan dapat menghadirkan tantangan teknologi tersendiri. Hal ini disebabkan oleh dinamika lingkungan dan lebih rawannya ancaman balik dari para penyelundup ilegal maupun para pencuri ikan illegal. Terlebih kapal-kapal penyelundup yang terbuat dari bahan non logam, akan sangat sulit diidentifikasikan jika menggunakan radar saja, sehingga dibutuhkan pengamatan visul scara langsung untuk mengetahui kondisinya.

Gambar II.3 C Sweep ASV
(Sumber : www.asvglobal.com)

Contohnya adalah ASV Global yang mengembangkan USV dengan kode C sweep untuk diaplikasikan dalam berbagai bidang seperti anti ranjau, anti kapal selam, dan pengintaian. Kendaraan ini memiliki kemampuan bermanuver dengan baik, *towing*, dan ketahanan komponen elektronik yang membuatnya dapat digunakan sebagai pendukung peperangan ranjau. Selain itu, kendaraan ini juga mampu menyebarkan, *tracking*, dan saling

terhubung dengan sistem autopilot lainnya baik USV, ROV, maupun AUV. Kendaraan ini dapat dijalankan dalam metode manual maupun dalam metode autopilot. Untuk mengendalikannya, kendaraan ini dikendalikan melalui darat menggunakan jaringan komunikasi berupa UHF (Ultra Gigh Frequance), L/S Band, maupun Sat-Comms Options Sidescan dan teknologi ASView untuk mengarahkannya. ASV tidak hanyamemiliki kemampuan untuk pengintaian saja, tapi juga dapat digunakan untuk pengejaran target. Untuk itu, kendaraan ini dilengkapi 2 mesin diesel dan kapasitas BBM yang besar yaitu 2300 liter sehingga memiliki kecepatan tinggi hingga 25+ knot serta dapat menjangkau area seluas 230 nm untuk ukuran kapal 10 meter.

Dengan ukurannya yang kecil, maka USV akan mudah dimobilisasi menggunakan kapal konvensional yang lebih besar. Pada prakteknya nanti, kapal konvensional ini dapat difungsikan sebagai kapal induk/pangkalan terapung yang membawahi beberapa USV. USV ini akan dilengkapi kamera pengintai yang akan digunakan untuk memonitor wilayah patrolinya. Kemudian USV ini disebar ke berbagai titik untuk melakukan patroli di wilayah yang telah ditetapkan sesuai koordinat GPS yang telah ditetapkan. Namun GPS memiliki keakuratan yang berbeda-beda tergantung lokasi dari titik koordinat tersebut. Selain itu, keadaan laut juga selalu berubah-ubah tergantung kondisi cuaca dan gelombang. (Didit, 2016)

II.1.3. Hukum Perbandingan

Purwarupa (bahasa Inggris: *prototype*) atau arketipe adalah bentuk awal (contoh) atau standar ukuran dari sebuah entitas. Dalam bidang desain, sebuah prototipe dibuat sebelum dikembangkan atau justru dibuat khusus untuk pengembangan sebelum dibuat dalam skala sebenarnya atau sebelum diproduksi secara massal. (Wikipedia)

Dalam memakai model fisik, hasil yang diperoleh harus ditransfer dari skala model ke skala penuh. Dengan demikian maka harus ada atau harus dinyatakan beberapa hukum perbandingan untuk keperluan transfer tersebut. Jika gaya spesifik yang bekerja pada model harus mirip dengan yang bekerja pada obyek yang berskala penuh maka syarat berikut perlu dipenuhi :

1. KesamaanGeometris

Dari segi permukaan, syarat kesamaan geometris biasanya diabaikan dan modelnya dibuat dengan permukaan yang benar-benar mulus. Pada kenyataannya, walaupun permukaan model dibuat persis menyerupai kapal yang sesungguhnya, aliran sepanjang permukaan

tersebut tidak akan mirip dengan aliran yang sebenarnya karena dipengaruhi oleh sifat air. Karena itu, hasil dari percobaan model harus dikoreksi.(Waka, 2012)

2. KesamaanKinematis

Rasio kecepatan pada model harus sama dengan rasio kapal skala penuh. Bila melakukan percobaan model baling-baling kapal, rasio antara kecepatan maju dengan kecepatanrotasional elemen daun baling-baling model harus sama dengan rasio kecepatan tersebut untuk baling-baling skala penuh. Rumus yang digunakan untuk rasio kecepatan kapal dengan kecepatan model adalah :

$Vm = Vs \times \sqrt{(Lm/Ls)}$

Dimana

Vm = Kecepatan model (knot)

Vs = Kecepatan kapal (knot)

Lm =Panjang model (meter)

Ls = Panjang kapal (meter)

(Waka, 2012)

3. Kesamaan Dinamis

Jika percobaan model yang dilakukandimaksudkan untuk mendapatkan informasi mengenai besarnya gaya yang bekerja pada kapal yang ditinjau, maka harus ada kesamaan dinamis. Antara model dan kapal dianggap terdapat kesamaan geometris dan kinematis. Selain itu dianggap bahwa :

$\lambda L = \frac{Ls}{Lm}$ = Skala panjang

$\lambda \rho = \frac{\rho s}{\rho m}$ = Skala massa jenis spesifik

$\lambda V = \frac{Vs}{Vm}$ = Skala kecepatan

dari sini diperoleh :

$\lambda s = \lambda L^2$ = skala permukaan

$\lambda \nabla = \lambda L^3$ = skala volume

$\lambda m = \lambda \rho \lambda L^3$ = skalamassa

$\lambda a = \frac{\beta V2}{\beta L}$ = Skala percepatan

(Waka, 2012)

II.1.4. Sistem Navigasi

Navigasi adalah ilmu pengetahuan dalam menentukan posisi kapal di laut dengan mengemudikan (steering) kapal secara aman dari suatu tempat ke tempat lain. Sistem navigasi biasanya terdiri dari beberapa perangkat digital maupun analog, untuk yang analog biasanya dilengkapi dengan kompas analog yang dapat mengetahui arah mata angin yang berguna sebagai acuan arah kapal, untuk perangkat digital sudah terdapat GPS atau *Global Positioning System* yaitu sebuah perangkat yang dapat menerima lokasi keberadaan kapal dengan mengacu pada satelit yang bergerak mengitari bumi. GPS menerima data yang dikirim dari satelit berupa data NMEA 0183. NMEA (*National Marine Electronics Association*) adalah standar yang digunakan dalam pengiriman data GPS yang berupa protokol data, garis lintang, garis bujur, ketinggian, dan waktu. Kompas digital juga tergolong perangkat digital dimana pemakaiannya harus diintegrasikan kembali pada sebuah sistem sehingga pembacaan arah mata angin dapat dilakukan dan dapat mengetahui arah kapal.(Sulistiyani, 2017)

II.1.5. Sistem Kontrol

Pengertian sistem kontrol itu sendiri adalah proses pengaturan / pengendalian terhadap satu atau beberapa besaran (variabel, parameter) sehingga berada pada suatu harga atau dalam suatu rangkuman harga (*range*) tertentu. Dalam istilah lain disebut juga teknik pengaturan, sistem pengendalian atau sistem pengontrolan. Secara umum sistem kontrol dapat dikelompokkan sebagai berikut :

1. Dengan operator (manual) dan otomatik.
2. Jaringan tertutup (*closed-loop*) dan jaringan terbuka (*open-loop*).
3. Kontinu (analog) dan diskontinu (digital, diskrit).
4. Servo dan regulator.
5. Menurut sumber penggerak : elektris, *pneumatis* (udara, angin), *hidarulis* (cairan), dan mekanis. (kontrol otomatik teori dan penerapan : 1994)

Sedangkan aksi pengontrolan ada enam aksi yaitu :

1. Dua posisi (on-off).
2. *Proportional*.
3. *Integral*.

4. *Proportional plus Integral*.
5. *Proportional plus Derivative.*
6. *Proportional plus Integral plus Derivative.*

Aksi kontrol PID (*Proportional, Integral, Derivative*) banyak ditemukan di dunia industri dan satu – satunya strategi yang paling banyak diadopsi pada pengontrolan proses. Berdasarkan survey, 97% industri yang bergerak dalam bidang proses (seperti kimia, *pulp*, makanan, minyak, dan gas) menggunakan PID sebagai komponen utama dalam pengontrolannya. (Browning, 1990)

Sistem kontrol PID (*Proportional-Integral-Derivative*) merupakan suatu kontroler yang berfungsi untuk menentukan kepresisian (kestabilan) suatu sistem instrumentasi dengan karakteristik umpan balik pada sistem tersebut. Kontroler PID terdiri dari tiga parameter yaitu P (*proportional*), D (*derivative*), I (*integral*). Dengan masing-masing parameter memiliki kelebihan dan kekurangan. Dalam implementasinya masing-masing parameter dapat bekerja sendiri maupun menggabungkan dari parameter tersebut. Parameter P, I dan D merupakan parameter yang diatur dalam sistem sesuai terhadap *input* sistem yang diinginkan. Sistem kontrol PIDbanyakdigunakanuntukpengaruhberbagaiprosesproduksi.

Sebuah kontroler PID secara kontinyu menghitung nilai kesalahansebagai beda antara *setpoint* yang diinginkan dan variabel prosesterukur. Kontroler mencoba untuk meminimalkan nilai kesalahan setiap waktu dengan penyetelan variabel kontrol, seperti posisi keran kontrol, *damper*, atau daya pada elemen pemanas, ke nilai baru yang ditentukan oleh jumlahan:

$$u(t) = Kp.e(t) + Ki \int_0^t e(\tau)d\tau + Kd\frac{de(t)}{dt}$$

Keterangan :

Kp = *Gain proportional*
Ki = *Gain integral*
Kd = *Gain derivative*
e = *Error = Ysp – Ym*
Ysp = *Setpoint*
Ym = Variabel proses

t = Waktu

τ = Variabel integrasi, nilainya diambil dari waktu nol sampai t

Transfer fungsi dalam bentuk Domain Laplace kontroler PID adalah

$$L(s) = Kp + \frac{Ki}{s} + Kd.s$$

Keterangan :

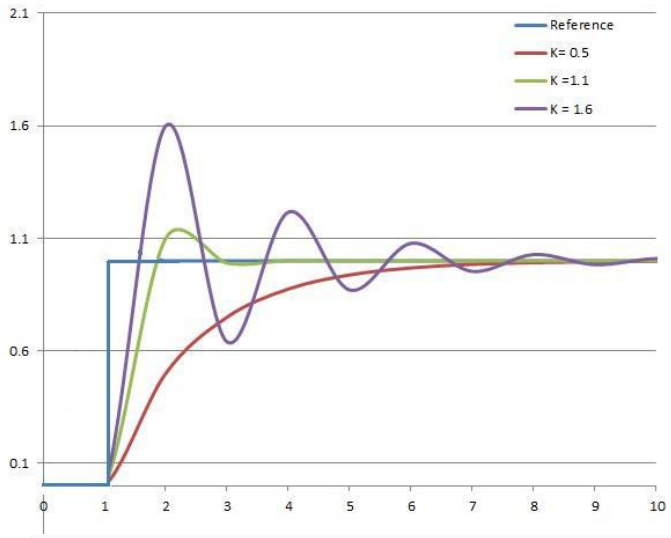
s = frekuensi bilangan kompleks

Dengan *Kp*, *Ki*, dan *Kd*, semuanya positif, menandakan koefisien untuk term *proportional*, *integral*, dan *derivatif*, secara berurutan (atau P, I, dan D). (Browning, 1990)

- *Term proportional* bertanggung jawab untuk nilai kesalahan saat ini. Contohnya, jika nilai kesalahan besar dan positif, maka keluaran kontrol juga besar dan positif. Dapat dirumuskan :

  $$P\ out = Kp.e(t)$$

  Gain yang besar menghasilkan perubahan yang besar pada keluaran untuk suatu nilai kesalahan tertentu. Namun, jika g*ain* terlalu besar, sistem dapat menjadi tidak stabil (lihatGambar II.4). Sebaliknya, gain yang bernilai kecil maka respon keluaran juga kecil, sehingga kontroler menjadi kurang responsif/sensitif, akibatnya tindakan kontrol menjadi terlalu kecil bila ada gangguan.

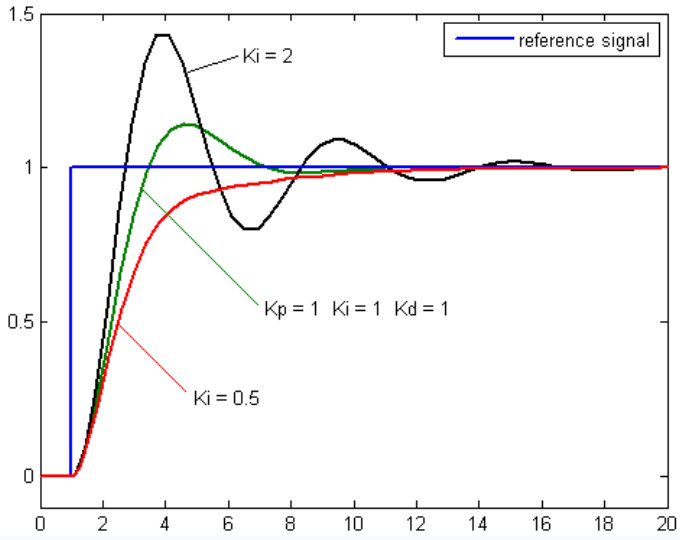


Gambar II.4Plot Y vs waktu, untuk 3 nilai K_p (K_i dan K_d dijaga konstan)
(Sumber : www.wikipedia.com)

- *Term integral* bertanggung jawab untuk nilai kesalahan sebelumnya. Contoh, jika keluaran saat ini kurang besar, maka kesalahan akan terakumulasi terus menerus, dan kontroler akan merespon dengan keluaran lebih tinggi. Dapat dirumuskan

$$I\ out = Ki \int_0^t e(\tau) d\tau$$

Term integral mempercepat perpindahan proses menuju *setpoint* dan menghilangkan *steady-state error* yang muncul pada kontroler *proportional*(Lihat gambar II.5). Namun, karena integral merespon terhadap *error* terakumulasi dari sebelumnya, maka dapat menyebabkan *overshoot*.

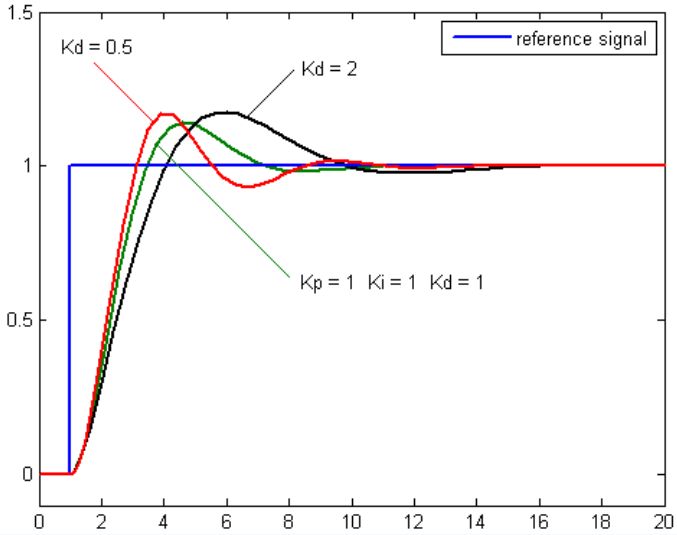


Gambar II.5Plot Y vs waktu, untuk 3 nilai K_i (K_p dan K_d dijaga konstan)
(Sumber : www.wikipedia.com)

- *Term Derivatif* bertanggung jawab untuk kemungkinan nilai kesalahan mendatang, berdasarkan pada rate perubahan tiap waktu (Lihat gambar II.6).Dapat dirumuskan :

$$D\ out = Kd\ \frac{de(\mathrm{t})}{dt}$$

Aksi derivatif memprediksi perilaku sistem dan kemudian memperbaiki waktu tinggal dan stabilitas sistem. Aksi derivatif jarang digunakan pada industri - diperkirakan hanya 25% kontroler karena akibatnya pada stabilitas sistem pada aplikasi dunia nyata.(Tim Wescott, 2000)



Gambar II.6Plot Y vs waktu, untuk 3 nilai K_d (K_p dan K_i dijaga konstan)
(Sumber : www.wikipedia.com)

Karena kontroler PID hanya mengandalkan variabel proses terukur, bukan pengetahuan mengenai prosesnya, maka dapat secara luas digunakan. Dengan penyesuaian (*tuning*) ketiga parameter model, kontroler PID dapat memenuhi kebutuhan proses. Respon kontroler dapat dijelaskan dengan bagaimana responnya terhadap kesalahan, besarnya *overshoot* dari *setpoint*, dan derajat osilasi sistem. Penggunaan algoritma PID tidak menjamin kontrol optimum sistem atau bahkan kestabilannya.

Kapal dapat dimodelkan sebagai badan kaku dengan enam derajat kebebasan. Namun untuk studi desain kapal permukaan autopilot, gerakan horizontal lambung kapal, yang didorong oleh baling-balingnya dan dikendalikan oleh kemudi adalah perhatian utama. Karena itu, gerakan yang dianggap hanya gerakan horizontal saja. (Browning, 1990)

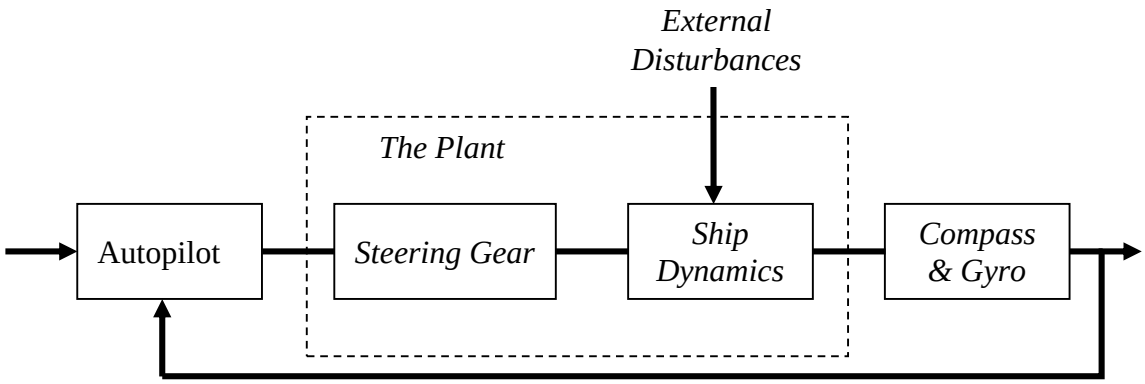
II.1.6. Sistem Autopilot

Sistem autopilot adalah sebuah sistem mekanikal, elektrikal, atau hidraulik yang memandu sebuah kendaraan tanpa campur tangan dari manusia. Umumnya autopilot dihubungkan dengan pesawat, tetapi autopilot juga digunakan di kapal dengan istilah yang sama.(www.wikipedia.com)

Pergerakan kapal dalam mode autopilot dipandu oleh sistem navigasi dan sistem kontrol. Sistem navigasikapalpada autopilotmenggunakan metode *waypoint*. Sistem navigasi akan memandu USV bergerak secara teratur mengikuti titik tuju (*waypoint*) yang telah diatur pada *Ground Control Station* (GCS). Dalam pergerakan USV, gaya luar

seperti angin dan ombak maupun gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya. Maka dari itu dibutuhkanlah sistem kontrol untuk mengendalikan kapal tetap menuju ke titik tujunya.

Dalam studi simulasi sistem kemudi USV, penting bahwa model dinamika sistem yang relatif akurat mempengaruhi kinerja kemudi.Sistem yang dimaksud bisa dianggap sebagai kapal dengan kemudiaktuator, yang mengalami gangguan eksternal seperti ditunjukkan pada Gambar II.7. (Lim, 1980)



Gambar II.7*Block Diagram of an Autopilot System*
(Sumber : Lim, 1980)

II.1.7. First Person View (FPV)

First Person View (FPV) atau dikenal juga dengan *Remote-person View* (RPV) merupakan metode yang digunakan untuk mengontrol sebuah wahana atau kendaraan *radio control* dari sudut pandang pilot. Sebagian besar FPV digunakan untuk wahana udara tak berawak (UAV) atau pesawat yang memakai *radio control*. Dengan kamera yang diletakkan tersebut kita dapat merasakan seolah-olah kita berada di dalam wahana tersebut dan melakukan pengendalian wahana dengan mudah. Pergerakan wahana tetap dikendalikan oleh operator secara manual, dengan adanya FPV maka operator dapat mengetahui arah, kondisi sekitar maupun lokasi yang akan dituju.(DroneZon.com)

II.2. Tinjauan Pustaka

Berisi referensi dan/atau hasil penelitian terdahulu yang relevan yang digunakan untuk menguraikan teori, temuan, dan bahan penelitian atau desain lain yang diarahkan untuk menyusun kerangka pemikiran atau konsep yang akan digunakan dalam penelitian.

II.2.1. Sistem Propulsi

Sistem propulsi ini memungkinkan agar USV dapat bergerak dan bermanuver dengan baik. Sistem propulsi USV ini menggunakan alat penggerak berupa Motor DC *brushless*, *Electronic Speed Control***(**ESC), baterai,motor Servo, *rudder*, dan *propeller* (baling-baling).

1. Motor *Brushless*DC

   *Brushless Motor*DCadalah jenis motor yang memiliki konstruksi magnet permanen dan sebuah stator berkutub yang dililit kawat. Energi listrik diubah menjadi energi mekanik oleh pengaruh daya tarik menarik antara kekuatan magnet permanen dengan stator berkutub yang dililit kawat tembaga. (Ogata, 1997)

   Bagian-bagian *brushless motor* adalah sebagai berikut:

   - Stator

     Dasar dari *brushlessmotor* adalah sebuah stator dengan memiliki tiga buah gulungan.

   - Rotor

     Rotor pada *brushless motor* terdiri dari beberapa magnet permanen. Jumlah kutub magnet di rotor juga mempengaruhi ukuran langkah dan riak torsi dari motor. Jumlah kutub yang banyak akan memberikan gerakan presisi dan riak torsi yang kecil. Magnet permanen terdiri dari 1 sampai 5 pasang kutub. Dalam kasus tertentu bisa 8 pasang kutub.

Gambar II.8*Motor Brushless* DC
(Sumber : https://hobbyking.com)

Motor brushless DCmemiliki banyak kelebihan dibanding motor brush, antara lain :

- Rugi-rugi daya kecil dikarenakan tidak adanya kontak *Brush* dengan rotor sehingga tidak terdapat gesekan.
- Kecepatan lebih tinggi.
- Motor mudah dikendalikan dengan menggunakan ESC.

Ada sejumlah istilah khusus terkait dengan motor *brushless*. Berikut adalah penjelasan untuk beberapa yang paling umum:

- RPM, yaitu ukuran kecepatan sudut, atau seberapa cepat sesuatu berputar. RPM motor berarti seberapa cepat motor itu bisa berputar.
- KV rating. Hubungan antara tegangan, torsi, dan RPM adalah linear untuk motor *brushless*. Jumlah RPM yang disediakan oleh masing-masing volt adalah sama, disebut nomor KV. Nomor KV ini berguna karena kita menjadi tahu berapa banyak volt yang kita perlukan untuk mencapai RPM tertentu, atau sebaliknya. Sebagai contoh, motor 980 KV didukung oleh baterai 11,1 volt akan berputar di 980 x 11,1 = 10.878 RPM tanpa beban. Rating KV selalu mengasumsikan tidak ada beban pada motor, sehingga RPM aktual yang dicapai akan kurang dari yang kita hitung.
- *Continuous / Burst Current*. *Continuous current* adalah berapa banyak arus yang dapat motor tangani secara terus menerus, untuk jangka waktu tertentu. Sedangkan *burst current* adalah berapa banyak arus motor dapat tangani dalam waktu singkat, atau sekitar beberapa detik.
- *Current rating* - adalah arus maksimum yang dapat motor tangani, diukur dalam ampere.
- *Inrunner / Outrunner*, yaitu desain motor *brushless*. Motor *brushless inrunner* memiliki kumparan stasioner, dan magnet permanen yang berputar dalam kumparan pada poros motor. Motor *brushless outrunner* adalah sebaliknya, ia memiliki magnet permanen yang berputar, ditempatkan di luar kumparan stasioner pada poros motor. Motor *outrunner* memiliki tingkat KV yang rendah, sehingga berputar pada kecepatan rendah dengan torsi lebih. Sehingga motor outrunner dapat secara langsung memutar *propeller* besar tanpa *gearbox*.
- Torsi, yaitu ukuran kekuatan sudut, atau berapa besar gaya dorong yang dimiliki oleh putaran poros motor.

2. *Electronic Speed Control* (ESC)

Motor *brushless* memiliki sebuah *Electronic Speed Control* (ESC) yang berfungsi sebagai pengatur kecepatan motor, selain itu juga berfungsi untuk menaikan-turunkan jumlah arus yang diperlukan oleh motor. Kecepatan untuk motor yang keluar dari ESC diatur melalui pulsa dari mikrokontroler atau *receiver*yang dikirim dari*Ground Control Station* (GCS). (Ogata, 1997)

Gambar II.9*Electronic Speed Control* (ESC)
(Sumber : http://www.hobbywing.com)

3. Baterai LiPo

Baterai LiPo yaitu jenis baterai yang tidak menggunakan cairan sebagai elektrolit melainkan menggunakan elektrolit polimer kering yang berbentuk seperti lapisan plastik film tipis. Lapisan film ini disusun berlapis-lapis diantara anoda dan katoda yang mengakibatkan pertukaran ion. Dengan metode ini baterai LiPo dapat dibuat dalam berbagai bentuk dan ukuran. Diluar dari kelebihan arsitektur baterai LiPo, terdapat juga kekurangan yaitu lemahnya aliran pertukaran ion yang terjadi melalui elektrolit polimer kering. Hal ini menyebabkan penurunan pada *charging* dan *discharging rate*. Masalah ini sebenarnya bisa diatasi dengan memanaskan baterai sehingga menyebabkan pertukaran ion menjadi lebih cepat, namun metode ini dianggap tidak dapat untuk diaplikasikan pada keadaan sehari-hari. (www.musbikhin.com)

Gambar II.10 Baterai LiPo
(Sumber : http://buaya-instrument.com)

4. Motor Servo

Motor servo adalah sebuah motor DC dengan sistem umpan balik tertutup di mana posisi rotor-nya akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor DC, serangkaian *gear*, potensiometer, dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut dari putaran motor servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor servo.

Secara putaran terdapat dua jenis motor servo, yaitu motor servo standar dan motor servo *continuous*. Motor servo standar sering dipakai pada sistem robotika, misalnya untuk membuat *Robot Arm* (Robot Lengan) sedangkan motor servo

continuous sering dipakai untuk *Mobile Robot*. Pada bagian servo tertulis tipe servo yang bersangkutan. Penggunaan motor servo di bidang robotika, yaitu motor servo yang memiliki putaran yang lambat dengan torsi yang kuat (berkat adanya sistem roda gigi). (Hutajulu, 2010)

Gambar II.11 Motor Servo
(Sumber : http://lapantech.com)

5. *Rudder*

Kemudi kapal merupakan suatu alat kapal yang digunakan untuk mengubah dan menentukan arah gerak kapal, baik arah lurus maupun belok kapal. Kemudi kapal ditempatkan pada ujung bagian belakang lambung kapal/buritan di belakang *propeller* kapal. Prinsip kerja kemudi kapal yaitu dengan mengubah arah arus fluida yang mengakibatkan perubahan arah kapal. Cara kerja kemudi kapal yaitu kemudi digerakkan secara mekanis atau hidrolik dari anjungan dengan menggerakkan roda kemudi.

Gambar II.12 Rudder
(Sumber : https://ae01.alicdn.com)

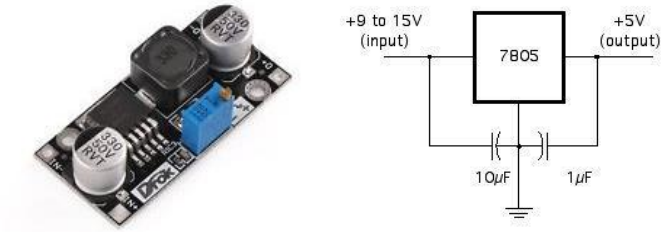
6. *Propeller*

Propeller merupakan sekelompok sayap berputar yang dibentuk bengkok, yang ditujukan agar menciptakan arah dari resultan gaya angkat yang menuju ke depan. Pada umumnya *propeller* terdiri dari dua atau lebih baling yang dihubungkan ke central hub yang merupakan bagian dimana baling – baling pesawat tersambung. *Propeller* berfungsi untuk mengubah gaya rotasi dari mesin menjadi gaya propulsif sebagai gaya dorong (*Thrust*) untuk kapal/pesawat. (Kroes, 1994).

Gambar II.13 *Propeller*
(Sumber : https://www.bazaargadgets.com)

7. Regulator

Regulator adalah komponen untuk untuk mengubah tegangan baterai dari 12 volt ke 5 volt. Baterai yang tersedia mamiliki tegangan 12 volt, dimana tegangan 12 volt terlalu besar untuk motor servo yang hanya memiliki tegangan 5 volt sehingga dibutuhkan alat untuk mengubah tegangan 12 volt menjadi 5 volt.



Gambar II.14 Regulator
(Sumber : http://thumbs1.ebaystatic.com)

II.2.2. Sistem Autopilot

Sistem autopilot ini memungkinkan agar USV dapat bergerak dan bermanuver sesuai dengan navigasi GPS dan kompas dengan baik. Sistem autopilot USV ini didukung komponen berupaMikrokontroler ArduPilot Mega, *Mission Planner*, baterai,modul GPS,kompas, dan *power bank.*

1. Mikrokontroler

Mikrokontroler adalah sistem mikroprosesor lengkap yang terkandung di dalam sebuah chip. Mikrokontroler berbeda dari mikroprosesor serba guna yang digunakan dalam sebuah PC, karena di dalam sebuah mikrokontroler umumnya juga telah berisi komponen pendukung sistem minimal mikroprosesor, yakni prosesor, memori dan antarmuka I/O, sedangkan di dalam mikroprosesor umumnya hanya berisi CPU saja. Berbeda dengan CPU serba-guna, mikrokontroler tidak selalu memerlukan memori eksternal, sehingga mikrokontroler dapat dibuat lebih murah dalam kemasan yang lebih kecil dengan jumlah pin yang lebih sedikit.

Mikrokontroler dapat menerima sinyal input, mengolahnya dan memberikan sinyal output sesuai dengan program yang diisikan ke dalamnya. Sinyal input

mikrokontroler berasal dari sensor yang berupa informasi sedangkan sinyal output ditujukan kepada aktuator yang dapat memberikan efek pergerakkan. Jadi secara sederhana mikrokontroler dapat diibaratkan sebagai otak dari suatu perangkat yang mampu berinteraksi dengan lingkungan sekitarnya. Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang di dalamnya terdapat mikroprosesor, memori, antarmuka Input/Output (I/O) dan perangkat pelengkap lainnya.

Kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan PC. Pada PC kecepatan mikroprosesor telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler berkisar antara 1 – 16 MHz. Begitu juga kapasitas RAM dan ROM pada PC yang bisa mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte. Mikrokontroler standar yang tersusun atas komponen-komponen sebagai berikut :

- *Central Processing Unit* (CPU)

  Central Processing Unit (CPU) merupakan bagian utama dalam suatu mikrokontroler. CPU pada mikrokontroler ada yang berukuran 8 bit ada pula yang berukuran 16 bit. CPU ini akan membaca program yang tersimpan di dalam ROM dan melaksanakannya.

- *Read Only Memory* (ROM)

  Read Only Memory (ROM) merupakan suatu memori yang sifatnya hanya dibaca saja. Dengan demikian ROM tidak dapat ditulisi. Dalam dunia mikrokontroler ROM digunakan untuk menyimpan program bagi mikrokontroler tersebut. Program tersimpan dalm format biner (‘0’ atau ‘1’). Susunan bilangan biner tersebut bila telah terbaca oleh mikrokontroler akan memiliki arti tersendiri.

- *Random Access Memory* (RAM)

  Random Access Memory (RAM) adalah jenis memori selain dapat dibaca juga dapat ditulis berulang kali. Tentunya dalam pemakaian mikrokontroler ada semacam data yang bisa berubah pada saat mikrokontroler tersebut bekerja. Perubahan data tersebut tentunya juga akan tersimpan ke dalam memori. Isi pada RAM akan hilang jika catu daya listrik hilang.

- *Input/Output* (I/O)

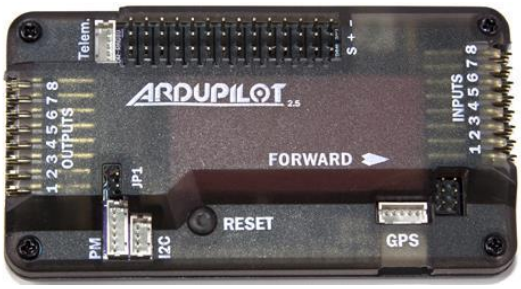
- Untuk berkomunikasi dengan dunia luar, maka mikrokontroler menggunakan terminal *Input/Output* (I/O), yang digunakan untuk masukan atau keluaran.

- Komunikasi Serial

  Komunikasi serial adalah salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu. Pada dasarnya komunikasi serial adalah kasus khusus komunikasi paralel dengan nilai n = 1, atau dengan kata lain adalah suatu bentuk komunikasi paralel dengan jumlah kabel hanya satu dan hanya mengirimkan satu bit data secara simultan. (Jim Turley, 2002)

2. ArduPilot Mega

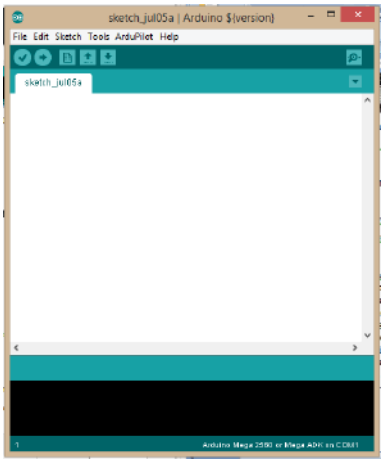
ArduPilot Mega adalah autopilot yang sepenuhnya diprogram dengan membutuhkan modul GPS dan sensor untuk membuat kendaraan berfungsi *Unmanned Aerial Vehicle* (UAV). Autopilot menangani baik stabilisasi dan navigasi, menghilangkan kebutuhan untuk sistem stabilisasi yang terpisah. Hal ini juga mendukung mode "*fly-by-wire*" yang dapat menjaga kestabilan wahana ketika beroperasi secara manual di bawah kendali *radio control*, sehingga lebih mudah dan lebih aman.Sensor yang digunakan GPS dan kompas. Penempatan GPS dan kompas ini bisa diletakkan pada *casing* atau wadah ArduPilot itu sendiri ataupun tempat yang lainnya.(Oborne, 2012).



Gambar II.15 ArduPilot Mega
(Sumber : http://www.ardupilot.co.uk)

3. *Software Arduino*

*Software Arduino*yang digunakan adalah driver dan IDE, walaupun masih ada beberapa software lain yang sangat berguna selama pengembangan arduino. IDE atau *Integrated Development Environment* merupakan suatu program khusus untuk suatu komputer agar dapat membuat suatu rancangan atau sketsa program untuk modul Arduino. (www.arduino.cc)



Gambar II.16*Software Arduino*

IDE arduino merupakan *software* yang ditulis dengan menggunakan bahasa C. IDE arduino terdiri dari:

- Editor program sebuah *window* yang memungkinkan pengguna menulis dan mengedit program dalam bahasa processing

- *Compiler* Sebuah modul yang mengubah kode program menjadi kode biner bagaimanapun sebuah mikrokontroler tidak akan bisa memahami bahasa *processing*.

- *Uploader* Sebuah modul yang memuat kode biner dari komputer ke dalam memori didalam modul arduino.

4. *Software Mission Planner*

Mission Planner adalah aplikasi yang digunakan untukmelakukan pemrograman pada ArduPilot *open source* untuk sistem autopilot pada beragam kendaraan *autonomous.*. *Mission Planner* digunakan untuk memantau pergerakan USV dari *Ground Control Station* (GCS). Aplikasi ini hanya dapat digunakan untuk Windows saja. (www.ardupilot.org)

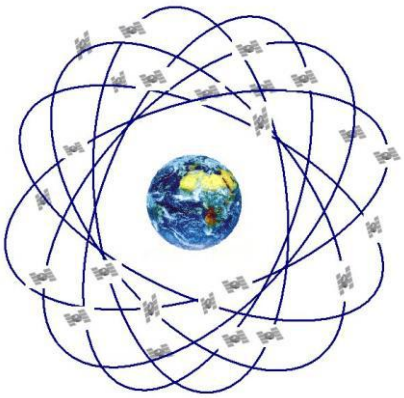
Gambar II 1 Tampilan *Mission Planner*

5. SensorGPS

GPS (*Global Positioning System*) adalah sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan letak, kecepatan, arah, dan waktu. Sistem yang serupa dengan GPS adalah GLONASS Rusia, Galileo Uni Eropa, IRNSS India. (www.wikipedia.com)

Sistem ini didesain untuk memberikan posisi dan kecepatan tiga-dimensi serta informasi mengenai waktu, secara kontinyu di seluruh dunia tanpa bergantung waktu dan cuaca, bagi banyak orang secara simultan.. Beberapa kemampuan GPS antara lain dapat memberikan informasi tentang posisi, kecepatan, dan waktu secara cepat, akurat, murah, dimana saja di bumi ini tanpa tergantung cuaca. Ketelitian dari GPS dapat bervariasi dari beberapa milimeter (orde nol) sampai dengan puluhan mater untuk ketelitian posisinya, beberapa cm/s untuk ketelitian kecepatannya dan beberapa nanodetik untuk ketelitian waktunya. Ketelitian posisi yang diperoleh akan tergantung pada beberapa faktor yaitu matode penentuan posisi, geomatri satelit, tingkat ketelitian data, dan matode pengolahan datanya.

Prinsip penentuan posisi dengan GPS yaitu menggunakan matode reseksi jarak, dimana pengukuran jarak dilakukan secara simultan ke beberapa satelit yang telah diketahui koordinatnya. Pada pengukuran GPS, setiap epoknya memiliki empat paramater yang harus ditentukan : yaitu 3 paramater koordinat X,Y,Z atau L,B,h dan satu paramater kesalahan waktu akibat ketidaksinkronan jam osilator di satelit dengan jam di *receiver* GPS. Oleh karena diperlukan minimal pengukuran jarak ke empat satelit. (Dietrich, 2013)



Gambar II.17 Orbit Stelit GPS Mengelilingi Bumi
(Sumber : aliamirrudin.files.wordpress.com)

Modul penerima GPS berukuran kecil serta memiliki prosesor dan antena yang secara langsung menerima data yang dikirim oleh satelit dan menghitung posisi dan waktu dengan cepat. Modul penerima GPS menggunakan konstelasi satelit dan stasiun bumi untuk menghitung posisi dan waktu hampir di mana saja di bumi.Pada setiap waktu, setidaknya ada 24 satelit aktif mengorbit lebih dari 12.000 mil di atas bumi. Posisi satelit diatur sedemikian rupa agar langit di atas lokasi kita akan selalu berisi paling tidak 12 satelit. Tujuan utama dari 12 satelit adalah untuk mengirimkan informasi kembali ke bumi melalui frekuensi radio (mulai 1,1 - 1,5 GHz). Dengan informasi ini modul penerima GPS dapat menghitung posisi dan waktu.

Jika antena penerima GPS dapat melihat setidaknya 4 satelit, modul GPS dengan secara akurat dapat menghitung posisi dan waktu. Akurasi GPS tergantung pada sejumlah variabel, terutama sinyal untuk rasio kebisingan sinyal, posisi satelit, cuaca dan penghalang seperti gedung dan pegunungan. Faktor-faktor ini dapat membuat kesalahan dalam penunjukkan lokasi yang didapatkan. Noise pada sinyal biasanya menciptakan kesalahan dari sekitar satu sampai sepuluh meter. Pegunungan, bangunan dan hal-hal lain yang mungkin menghalangi jalan antara penerima dan satelit dapat menyebabkan tiga kali lebih banyak kesalahan daripada noise pada sinyal. Sebuah penerima GPS harus bisa dapat mengunci pada 4 satelit untuk dapat memperoleh posisi. Meskipun dimungkinkan untuk mendapatkan posisi pada kurang dari 4 satelit, *error* dari posisi yang didapat bisa menjadi agak besar. Pembacaan paling akurat bisa didapat ketika langit cerah, jauh dari penghalang dan mengunci pada lebih dari 4 satelit. Untuk menanggulangi masalah ini, beberapa asisten yang berbeda telah diciptakan seperti *Assisted* GPS dan *Differential*GPS.

Kebanyakan modul GPS memiliki *port serial,* yang membuatnya cocok untuk terhubung ke mikrokontroler atau komputer.Setelah modul GPS diaktifkan, data dengan format NMEA (atau format lain) dikirim keluar dari pin *erial transmit* (TX) dengan *baud rate* dan update *rate* tertentu, bahkan ketika tidak ada lock. Agar mikrokontroler dapat membaca data NMEA dari modul GPS, dilakukan dengan cara menghubungkan pin TX (pengirim) pada GPS ke pin RX (penerima) pin ada mikrokontroler. Untuk mengkonfigurasi modul GPS, dilakukan dengan cara menghubungkan pin RX pada GPS ke pin TX pada mikrokontroler. Pada mikrokontroler data NMEA dari modul GPS diuraikan dengan cara menghapus

potongan data dari entence NMEA, sehingga mikrokontroler dapat melakukan sesuatu yang berguna dengan data tersebut.

Gambar II.18 Modul GPS
(Sumber : dna3.zoeysite.com)

6. Sensor Kompas dan Giroskop

Kompas adalah alat navigasi untuk menentukan arah berupa sebuah panah penunjuk magnetis yang bebas menyelaraskan dirinya dengan medan magnet bumi secara akurat. Kompas memberikan rujukan arah tertentu, sehingga sangat membantu dalam bidang navigasi. Arah mata angin yang ditunjuknya adalah utara, selatan, timur, dan barat. Apabila digunakan bersama-sama dengan jam dan sekstan, maka kompas akan lebih akurat dalam menunjukkan arah. Alat ini membantu perkembangan perdagangan maritim dengan membuat perjalanan jauh lebih aman dan efisien dibandingkan saat manusia masih berpedoman pada kedudukan bintang untuk menentukan arah. Dalam kompas, medan magnet yang diukur adalah medan magnet bumi yang mengalir dari utara ke selatan. Sensor GPSmemiliki tiga sumbu yang berbeda untuk menghitung arah, tetapi faktor kemiringan tidak diperhitungkan maka digunakan hanya sumbu X dan Y saja.(Dietrich, 2013)

Sensor giroskop atau gyros, adalah perangkat yang mengukur atau mempertahankan gerak rotasi. Gyros jenis MEMS (*microelectromechanical sistem*) berbentuk kecil dan murah yang dapat mengukur kecepatan sudut. Satuan kecepatan sudut diukur dalam derajat per detik (°/s) atau putaran per detik (RPS). Kecepatan sudut merupakan hasil pengukuran kecepatan rotasi. Giro dapat digunakan untuk menentukan orientasi atau arah dan banyak dipakai pada sistem navigasi otomatis. Sebagai contoh, jika kita ingin menyeimbangkan robot, giroskop dapat digunakan untuk mengukur rotasi dari posisi yang diseimbangkan dan mengirim koreksi untuk motor.

Giro sering digunakan pada objek yang tidak berputar dengan sangat cepat. Sebagai contoh pesawat, pesawat tidak berputar akan tetapi pesawat hanya memutar beberapa derajat pada setiap sumbu. Dengan mendeteksi perubahan kecil pada sudut tersebut gyros membantu menstabilkan penerbangan pesawat. Percepatan atau kecepatan linear pesawat tidak mempengaruhi pengukuran gyro. Gyros hanya mengukur kecepatan sudut. Sensor giroskop MEMS sangat kecil (antara 1 sampai 100 mikrometer, seukuran rambut manusia). Ketika gyro diputar, massa beresonansi kecil digeser sebagai perubahan kecepatan sudut. Gerakan ini diubah menjadi sinyal-sinyal listrik yang sangat rendah yang dapat diperkuat dan dibaca oleh mikrokontroler. Pada modul APM sudah terdapat sensor *gyros internal.* (www.arduino.cc)

7. Power Bank

Power Bank adalah pengisi daya gadget saat sedang berada diluar dan jauh dari sumber listrik yang bersifat portable. Fungsi power bank dapat disebut juga sebagai penyimpan daya atau dapat dianalogikan sebagai baterai cadangan, namun untuk penggunannya kita tidak perlu melepas baterai dari perangkat, namun cukup menacapkan kabel USB seperti saat kita mengisi daya menggunakan charger biasa.

Gambar II.19 Power Bank
(Sumber : http://energonacc.com)

II.2.3. Sistem Komunikasi *Wireless*

Sistem komunikasi *wireless* merupakan sistem penyampaian informasi (berupa data,suara, gambar, video) tanpa menggunakan media kabel sebagai perantara, tetapi menggunakan media yang lain berupa udara yang dibawa lewat gelombang. Dalam sistem komunikasi *wireless* ada bagian umum gelombang yang berperan menjadi bagian utuh untuk komunikasi, yaitu : gelombang radio, gelombang mikro, gelombang infra merah dan gelombang elektromagnetik. Di dalam sistem komunikasi *wireless* dengan frekuensi radio terdiri dari perangkat-perangkat data, *transmitter* dan *receiver*.(www.groupjono.wordpress.com)

1. **Data**

Data dalam komunkasi *wireless* ini bisa berupa video, audio, dan data-data yang lain. Data yang masuk sebagai input analog akan diubah menjadi data digital laluditransmisikan dan diterima *receiver,* berikutnya akan diubah dari data digital menjadi data analog.

2. ***Transmitter* dan *Receiver***

Transmitter adalah bagian dari sistem komunikasi *wireless* yang berfungi untuk mengirimkan data ke tempat lain berupa gelombang radio. *Receiver* merupakan bagian yang berfungsi untuk menerima sinyal atau data yang dikirimkam oleh *transmitter*.(www.groupjono.wordpress.com)

Contohnya adalah *Trasmitter* dan *Receiver Radio Control* 2.4 GHz yang berfungsi untuk mengendalikan USV dalam mode manual dan untuk mengaktifkan *switch* mode autopilot. Selain itu adalah *Radio Telemetry* 915 MHz yang berfungsi untuk menghubungkan ArduPilot Mega pada USV dengan *Mission Planner* pada *Ground Control Station* (GCS). Kemudian ada *Image Transmitter* untuk menghubungkan kamera pada USV dengan *Ground Control Station* (GCS).

- *Radio Control*

  Terdapat dua stik utama untuk mengontrol *throttle* dan arah, serta beberapa *switch*, yang dapat digunakan untuk merubah mode.

  Throttle – Menambah/mengurangi kecepatan USV

  Yaw/Rudder – Membelokkan USV ke kanan atau kiri

Modul remot kontrol terdiri dari satu pasang, yaitu satu buah untuk dipasang pada USV dan satu buah untuk GCS. Modul ini beroperasi pada frekuensi 2.4 GHz. Memiliki jarak jangkau sekitar ±200 meter.

Gambar II.20 Trasmitter dan Receiver Radio Control
(Sumber : http://www.hooked-on-rc-airplanes.com)

- *Radio Telemetry*

  Modul telemetri kompatibel dengan modul APM. Modul ini dirancang sebagai *open source* dan memiliki jangkauan sekitar satu mil (atau sekitar 1.6 km). Modul telemetri yang dipakai pada tugas ini beroperasi pada frekuensi 915 MHz. Sistem telemetri yang digunakan terdiri dari modul telemetri untuk USV dan 1 buah modul telemetri USB untuk ground station (laptop). Modul ini menggunakan antarmuka melalui FTDI USB serial.

Gambar II.21*Radio Telemetry* 915 MHz
(Sumber : http://g03.a.alicdn.com)

II.2.4. Sistem Kamera

Kamera yang digunakan dapat berupa CCTV maupun kamera*action.* Kamera *action* adalah kamera yang biasanya digunakan di luar ruangan. Kamera ini memiliki dimensi yang kecil sehingga dapat dipindahkan dengan mudah. Kamera ini dapat dihubungkan dengan perangkat berupa handphone maupun tablet sehingga pengguna tidak harus memencet tombol yang ada pada kamera, tetapi dari handphone atau tablet. Pengiriman datanya dilakukan secara *wireless*. Keterbatasannya hanya ada pada jarak jangkauan antara kamera dan perangkat handphone atau tabletnya saja. Maka dari itu, pada sistem kamera USV, diberi tambahan berupa *image transmitter*.

Gambar II.22 Kamera Action
(Sumber : https://shop.gopro.com)

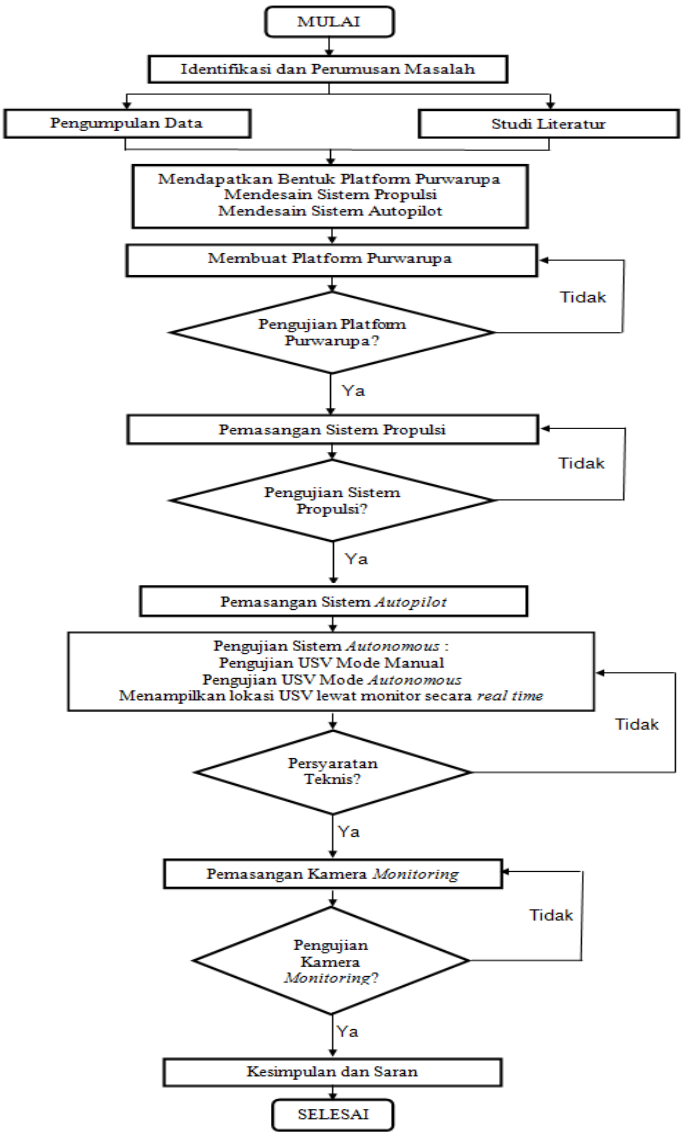
*Image transmitter*merupakan alat bantu untuk mengirimkan gambar jarak jauh. Dengan tambahan *image transmitter,* jarak jangkau kamera dapat ditambah hingga jarak ±2 km. *Image transmitter*dipasang di USV dan dihubungkan ke *action camera.* Kemudian sinyal hasil *image processing* akan dikirimkan ke *Ground Control Station* (GCS).

Gambar II.23*Image Transmitter*
(Sumber : http://img.dxcdn.com)

BAB III
METODOLOGI

III.1. Diagram Alir

Berikut adalah diagram alir pengerjaan Tugas Akhir yang ditunjukkan pada Gambar 3.1. Tugas Akhir ini dimulai dengan identifikasi perumusan masalah, yang dilanjutkan dengan pengumpulan data dan studi literatur. Pada proses pembuatan purwarupa USV,bagian-bagian yang dibuat meliputi platform, sistem propulsi, sistem autopilot, dan sistem kamera*monitoring*. Pada setiap bagian purwarupa yang telah selesai dikerjakan, akan dilakukan pengujian dan analisa hasil untuk diambil kesimpulan dan saran dari hasil pengujiannya.



Gambar III.1Diagram Alir

III.2. Tahap Pengerjaan

III.2.1. Tahap Identifikasi Masalah

Pada tahap awal ini dilakukan identifikasi permasalahan berupa:

1. Terbatasnya jumlah kapal patroli, jumlah personil, dan dana yang dimiliki oleh TNI AL.
2. Terjadinya bentrokan antara pihak TNI AL dengan pihak lain ketika berpatroli sehingga menimbulkan korban.

III.2.2. Tahap Studi Literatur

Pada tahap ini dilakukan studi literatur yang berkaitan dengan permasalahan pada Tugas Akhir ini. Studi literatur dilakukan untuk mendapatkan pengetahuan serta teori-teori yang berkaitan dengan Tugas Akhir ini, bisa dalam bentuk hasil penelitian sebelumnya agar bisa lebih memahami permasalahan dan pengembangan yang dilakukan. Studi yang dilakukan diantaranya:

- Cara pembuatan platform purwarupa kapal

  Perlu untuk diketahui bagaimana proses pembuatan platform purwarupa kapal serta proses pencetakan kapal dengan menggunakan bahan fiber. Sehingga dapat dibuat suatu platform purwarupa kapal yang sesuai dengan gambar *Lines Plan* dan dapat diketahui kebutuhan bahan yang akan digunakanpada kapal.
- Sistem Propulsi

  Perlu untuk diketahui desain sistem propulsi serta proses pembuatan jaringan sistem propulsi pada kapal.
- Sistem Autopilot

  Perlu untuk diketahui desain sistem autopilotserta proses pembuatan jaringan sistem navigasi pada kapal.
- Sistem Komunikasi *Wireless*

  Perlu untuk diketahui desain sistem komunikasi serta proses pembuatan jaringan sistem komunikasi pada kapal.
- Sistem Kamera*Monitoring*

  Perlu untuk diketahui desain sistem kamera*monitoring*serta proses pembuatan jaringan sistem kamera*monitoring* pada kapal.

III.2.3. Tahap Pengumpulan Data

Metode pengumpulan data dalam Tugas Akhir ini adalah metode pengumpulan secara tidak langsung (sekunder). Pengumpulan data ini dilakukan dengan mengambil data terkait dengan permasalahan dalam tugas ini. Adapun data-data yang diperlukan antara lain:

- Ukuran Utama dan Bentuk Model Kapal

  Data mengenai ukuran utama dan bentuk model kapal didapatkan dari Tugas Akhir dari Dwiko Hardianto yang berjudul "Pembuatan Konsep Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia" dan Fajar Ramadhan yang berjudul "Pembuatan Detail Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia".. Dari data tersebut, akan dicari skala ukuran utama kapal dan bentuk model kapal untuk selanjutnya dibuat dalam ukuran model USV.

- Kebutuhan Sistem Propulsi

  Untuk spesifikasi komponen sistem propulsi yang akan digunakan nantinya akan diambil dari *data sheet*masing-masing komponen.

- Kebutuhan Sistem Autopilot

  Untuk spesifikasi komponen sistem autopilot yang akan digunakan nantinya akan diambil dari *data sheet*masing-masing komponen.

- Kebutuhan Sistem Komunikasi *Wireless*

  Untuk spesifikasi komponen sistem komunikasi *wireless* yang akan digunakan nantinya akan diambil dari *data sheet*masing-masing komponen.

- Kebutuhan Sistem Kamera*Monitoring*

  Untuk spesifikasi komponen sistem kamera*monitoring* yang akan digunakan nantinya akan diambil dari *data sheet*masing-masing komponen.

III.2.4. Tahap Tahap Desain USV

Tahap-tahap desain USV dalam Tugas Akhir ini antara lain:

- Mendapatkan Desain Rencana Garis dan Bukaan Kulit Platform Purwarupa

  Pembuatan rencana garis dilakukan dengan bantuan *software maxsurf Modeler Advance* dan bukaan kulit dengan bantuan *software maxsurf Workshop Professional.* Setelah proses desain rencana garis dan bukaan kulit selesai, proses berikutnya adalah menyempurnakan atau menyelesaikan desain rencana garis dengan bantuan *software AutoCad*.

- Mendapatkan Desain Sistem Propulsi

  Desain sistem propulsi dilakukan berdasarkan spesifikasi masing-masing komponen sistem propulsi berdasarkan *data sheet* yang tersedia.
- Mendapatkan Desain Sistem Autopilot

  Desain sistem autopilot dilakukan berdasarkan spesifikasi masing-masing komponen sistem autopilot berdasarkan *data sheet* yang tersedia.
- Mendapatkan Desain Sistem Kamera *Monitoring*

  Desain sistem kamera *monitoring*dilakukan berdasarkan spesifikasi masing-masing komponen sistem kamera *monitoring* berdasarkan *data sheet* yang tersedia.

III.2.5. Tahap Pembuatan USV

Tahap-tahap pembuatan USV dalam Tugas Akhir ini antara lain:

- Membuat Platform Purwarupa

  Pembuatan platform purwarupa dilakukan berdasarkan desain platform model USV yang telah selesai.
- Membuat Sistem Propulsi

  Pembuatan sistem propulsi dilakukan berdasarkan desain sistem propulsi yang telah selesai.
- Membuat Sistem Autopilot

  Pembuatan sistem autopilot dilakukan berdasarkan desain sistem autopilot yang telah selesai.
- Membuat Sistem Kamera *Monitoring*

  Pembuatan sistem kamera *monitoring* dilakukan berdasarkan desain sistem kamera *monitoring* yang telah selesai.

III.2.6. Tahap Pengujian dan Analisa USV

Tahap-tahap pengujian USV dalam Tugas Akhir ini antara lain:

- Pengujian dan Analisa Platform Purwarupa

  Pengujian dan analisa pada platform purwarupa mencakup kesesuaian ukuran utama dan bentuk *Lines Plan* purwarupa, tingkat kebocoran, daya apung, dan tingkat kestabilan.
- Pengujian dan Analisa Sistem Propulsi

  Pengujian dan analisa pada sistem propulsi mencakup kinerja motor induk, *propeller,* dan *rudder* pada purwarupa.
- Pengujian dan Analisa Sistem Autopilot

Pengujian dan analisa pada sistem autopilot mencakup pergerakan kapal pada mode manual dan autopilot, akses dan kalibrasi (GPS, kompas, dan *Mission Planner*), menampilkan lokasi USV pada layar monitor *Ground Control System* (GCS) secara *real time,* dan fungsi komunikasi.

- Pengujian dan Analisa Sistem Kamera *Monitoring*

  Pengujian dan analisa pada sistem kamera *monitoring* mencakup kinerja kamera yang digunakandan menampilkan citra gambar video kamera USV pada layar monitor *Ground Control System* (GCS) secara *real time*.

III.2.7. Kesimpulan dan Saran

Pada tahap ini dirangkum hasil perancangan, pembuatan, dan pengujian yang didapat dan saran untuk pengembangan lebih lanjut. Setelah semua tahapan selesai dilaksanakan, selanjutnya ditarik kesimpulan dari hasil pengujian dan analisa. Kesimpulan berupa ukuran utama purwarupa USV dan analisa dari hasil pengujian yang telah dilakukan meliputi pengujian platform, sistem propulsi, sistem autopilot, dan sistem kamera*monitoring*.

Saran dibuat untuk menyempurnakan terhadap beberapa hal yang belum tercakup di dalam proses desain ini.

III.3. Bahan dan Peralatann

III.3.1. Bahan

Bahan yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

2. Kertas HVS dan Duplex
3. Lem UHU, lem G, dan lem Epoxy
4. Mat Fiber
5. Resin dan katalis
6. Dempul dan katalis
7. Amplas 400, 800, dan 1000
8. Surfacer dan Cat
9. PVC Foam
10. Lakban dan *double tape*

III.3.2. Peralatan

Peralatan yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

1. Alat tulis
2. Gunting dan *Cutter*
3. Printer
4. Kuas
5. Gerinda dan *Sender*
6. Laptop dan Tablet

III.3.3. Komponen Sistem Propulsi

Komponen sistem propulsi yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

1. Motor DC *Brushless*
2. *Electronic Speed Control* (ESC)
3. *Propeller*
4. Poros dan *Shaft Tunnel*
5. *Rudder*
6. Motor Servo
7. Baterai LiPo 3S 12V 2200mAh

III.3.4. Komponen Sistem Autopilot

Komponen sistem autopilot yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

1. Mirokontroler ArduPilot Mega 2.8
2. Regulator
3. Modul GPS UBLOX M8N dan Kompas
4. *Radio Control* 2.4 Hz
5. *Radio Telemetry* 915 MHz
6. *Power Bank* Hippo 11000 mAh

III.3.5. Komponen Sistem Kamera *Monitoring*

Komponen sistem kamera *monitoring* yang digunakan dalam pengerjaan Tugas Akhir ini yakni:

1. Kamera *Action* GoPro Hero 4
2. 5.8G 600 MW 48 Channel AV
3. Baterai LiPo 2S 5V 660 mAh

BAB IV
DESAIN DAN PEMBUATAN

IV.1. Umum

Desain dan pembuatan purwarupa USV ini meliputi bebarapa bagian, antara lain sebagai berikut:

1. Pembuatan platform purwarupa, meliputi desain dan data *Unmanned Surface Vehicle* (USV) dan proses pembuatan platform purwarupa (lambung, dek, dan *main mast*)
2. Diagram blok sistem
3. Sistem Propulsi, meliputi desain dan pembuatan sistem propulsi
4. Sistem Autopilot, meliputi desain dan pembuatan sistem autopilot
5. Sistem Kamera *Monitoring*, meliputi desain dan pembuatan sistem kamera *monitoring*

IV.2. Pembuatan Platform Purwarupa

Proses pembuatan purwarupa USV dimulai dengan pembuatan platform purwarupa untuk mewadahi dan mengakomodasi bagian-bagian sistem propulsi, sistem autopilot, dan sistem kamera. Platform purwarupa dibuat berdasarkan desain *Lines Plan* dan bentuk model dari Tugas Akhir Dwiko Hardianto yang berjudul "Pembuatan Konsep Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia" dan Fajar Ramadhan yang berjudul "Pembuatan Detail Desain *Unmanned Surface Vehicle* untuk Monitoring Wilayah Perairan Indonesia".

Setelah didapat *Lines Plan* model, lalu dibuat cetakan kapal untuk menghasilkan lambung purwarupa USV. Lalu dilanjutkan pengerjaan dek dan *main mast* serta bagian-bagian akomodasi untuk sistem propulsi, sistem autopilot, dan sistem kamera*monitoring*. Setelah itu platform purwarupa siap diuji dan dilanjutkan ke proses berikutnya untuk pembuatan sistem propulsi, sistem autopilot, dan sistem kamera*monitoring*.

IV.2.1. Desain dan Data *Unmanned Surface Vehicle* (USV)

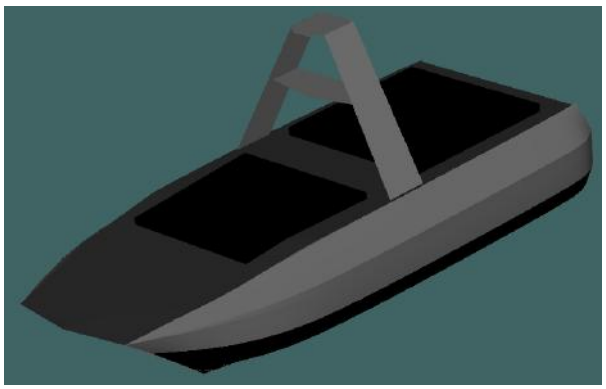
Data *Unmanned Surface Vehicle* (USV)yang akan digunakan pada pengerjaan Tugas Akhir ini adalah sebagai berikut :

- Jenis Kapal : *Unmanned Surface Vehicle* (USV)
- Panjang (Lwl) : 6.9 m

- Tinggi (H) : 1.15 m
- Lebar (B) : 3.5 m
- Sarat (T) : 0.5 m
- Kecepatan : 20 knot = 10.2889 m/s
- Displacement : 3.9 ton

Kapal tersebut akan dibuat purwarupa dalam skala kecil. Skala yang akan digunakan adalah 1 : 7 dengan menskalakan ukuran panjang mesin induk yang ada. Panjang mesin induk pada kapal asli adalah 1 meter sementara mesin induk yang tersedia untuk pembuatan purwarupa memiliki dimensi panjang 7 cm sehingga didapat perbandingan mesin induk kapal dan model adalah 1 : 7. Maka, dimensi purwarupa dibuat dalam skala 1 : 7 dari dimensi kapal sehingga didapat dimensi sebagai berikut :

- Panjang (LWL) : 98.6 cm
- Tinggi (H) : 16.5 cm
- Lebar (B) : 50 cm
- Sarat (T) : 7 cm
- Displacement : 9.78 kg
-



Gambar IV.1 Desain 3D USV

Untuk mencari kecepatan kapal, digunakan rumus :

Vm = Vs x $\sqrt{(Lm/Ls)}$

Dimana

Vm = Kecepatan model (knot)

Vs = Kecepatan kapal (knot)

Lm =Panjang model (meter)

Ls = Panjang kapal (meter)

Sehingga didapat kecepatan model :

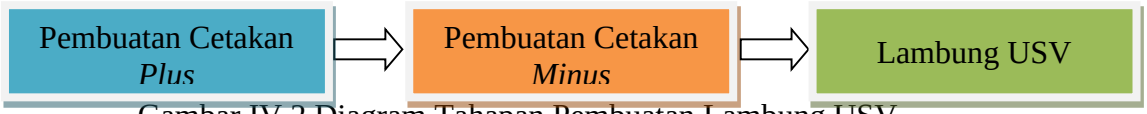
Vm = 12 x $\sqrt{0.986}/6.9$

Vm = 7.56 knot = 3.889 m/s

USV dapat dibuat menggunakan berbagai jenisbahan seperti fiber, kayu balsa, dan triplek. Untuk purwarupa kali ini, USV dibuat menggunakan bahan fiber. Pertimbangannya adalah faktor keamanannya karena bahannya lebih kuat dan komponen-komponen yang ada di dalam kapal bernilai tinggi. Selain itu, penulis lebih berpengalaman dalam pembuatan model kapal dengan menggunakan bahan fiber.

IV.2.2. Proses Pembuatan Platform Purwarupa

Proses pembuatan purwarupa pada Tugas Akhir ini diawali dengan pembuatan lambung. Pembuatan lambung kapal menggunakan 2 kali pencetakan untuk menghasilkan cetakan *plus* dan *minus*. Dari 2 kali pencetakan ini akan didapatkan lambung kapal sesungguhnya. Setelah menghasilkan bagian lambung sesungguhnya, pengerjaan dilakukan membuat bagian dek dan *main mast*. Setelah kedua bagian disambung, maka dilakukan *finishing* berupa penghalusan dan pewarnaan badan kapal.



Gambar IV.2 Diagram Tahapan Pembuatan Lambung USV

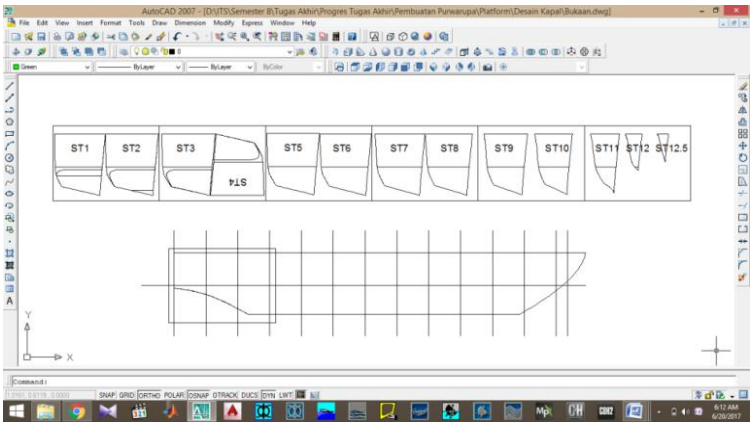
1. Proses Pembuatan Lambung

Tahap pertama adalah pembuatan lambung. Pembuatan lambung dimulai dengan pembuatan cetakan *plus*, lalu pembuatan cetakan *minus*, dan pembuatan platform lambung USV.

1. Tahapan Pembuatan Cetakan *Plus*:
   - Pencetakan gambar *body plan* dan kulit luar

     Pada proses ini, *body plan* didapat dari hasil tugas akhir sebelumnya yang membahas tentang konsep desain USV, kemudian dirapikan dengan AutoCAD. Pada pencetakan gambar *body plan* ini, gambar terlebih dahulu diskala ke bentuk model purwarupa yaitu 1 : 7. Lalu gambar *body plan* dipisah-pisah menjadi gambar tiap *station*, Lalu dicetak ke kertas HVS.

Untuk gambar bukaan kulit luar didapat dari software *maxsurf workshop professional*. Pada pencetakan gambar kulit luarini, yang digunakan adalah kulit luar bagian sisi luar dan sisi dalam lambung. Selanjutnya gambar dicetak ke kertas HVS.



Gambar IV.3 Pencetakan Gambar Lines Plan dan Kulit Luar

- Penyusunan rangka dan kulit lambung

Setelah gambar *station* dan kulit dicetak di HVS, kemudian gambar di kertas HVS ditempel di kertas duplex. Setelah itu kertas duplex dipotong mengikuti kurva yang tercetak di gambar kertas HVS. Selanjutnya, potongan *station* dan kulit kapal disusun hingga membentuk kerangka bagian lambung kapal.

Gambar IV.4 Penyusunan Rangka dan Kulit Lambung

- Pelapisan resin pada bagian luar

Lambung kapal yang sudah terbentuk awalnya masih belum kaku karena kertas duplex masih bersifat sedikit lentur. Selanjutnya, menyiapkan resin dan katalis yang akan digunakan untuk melapisi lambung kapal. Pelapisan oleh katalis

ini bertujuan untuk menguatkan kertas duplex. Jumlah campuran katalis yang digunakan umumnya sebesar 5% dari jumlah resin.

Gambar IV.5 Resin dan Katalis

- Penghalusan badan luar cetakan *plus*

Setelah lapisan resin kering, maka badan kapal didempul untuk melapisi celah-celah cetakan dan untuk meratakan cetakan agar saat digunakan menjadi rata dan tidak memiliki celah. Setelah kering, proses dilanjutkan dengan mengampas badan kapal. Pengamplasan adalah proses penghalusan bagian yang sudah didempul untuk mendapatkan hasil yang baik. Pertama siapkan amplas dengan ukuran 400. kemudian amplas bagian yang sudah di dempul. Jika pengamplasan sudah halus dan merata, lakukan pencucian dengan air sabun lalu gosokan dengan amplas yang ukuranya 1000. Gosokan berulang-ulang hingga halus agar mendapatkan hasil yang baik. Setelah pencucian dengan air sabun colek selesai, bersikan dengan kanebo atau majun yang kering.Setelah cetakan *plus* badan kapal telah selesai, proses ini dilanjutkan dengan pembuatan cetakan *minus.*

2. Tahapan Pembuatan Cetakan *Minus*:

- Pemberian Wax

Pemberian wax pada tiap-tiap bagian yang akan digunakan untuk mencetak semua bagian kapal. Pemberian wax ini dengan dengan cara digosok dengan menggunakan majun atau kain sampai merata dan sampai terlihat licin. Tujuannya agar pada saat badan kapal dilepas tidak terlalu melekat dan kering dengan cetakan sehingga hasil cetakan menjadi halus.

- Pelapisan dengan bahan fiber

Pelapisan semua bagian cetakan dengan bahan fiber yang terdiri dari mat , resin ,dan katalis. Mat dibentangkan hingga menutup seluruh bagian lambung kapal, kemudian diolesi dengan resin yang sudah tercampur dengan katalis. Kemudian resin diratakan dan ditekan-tekan menggunakan kuas agar resin bisa menyebar rata di seluruh permukaan mat. Sebelum kering, proses pelapisan ini diulang lagi untuk menambah lapisan bahan fiber hingga lapisan fiber mencapai 3 lapis. Hal ini dilakukan agar cetakan *minus* kuat dan tidak mudah pecah ketika pemisahan dengan badan kapal yang sesungguhnya nanti.

- Pemisahan antara cetakan *plus* dan cetakan *minus*

Setelah resin sudah benar-benar kering, dilakukan pemisahan antara cetakan *plus* dan cetakan *minus.* Karena yang akan digunakan hanya cetakan *minus* nya saja, maka cetakan *plus* dapat dihancurkan ketika dipisah dari cetakan *minus* nya sehingga disisakan cetakan *minus* nya saja. Kemudian cetakan ini dibersihkan dari sisa-sisa bekas cetakan *plus*.

- Penghalusan badan dalam cetakan

Setelah cetakan *minus* bersih, kemudian dilanjutkan proses pendempulan untuk melapisi celah-celah cetakan dan untuk meratakan cetakan agar saat digunakan menjadi rata dan tidak memiliki celah. Setelah kering, proses dilanjutkan dengan mengampas badan kapal. Pengamplasan adalah proses penghalusan bagian yang sudah didempul untuk mendapatkan hasil yang baik. Pertama siapkan amplas dengan ukuran 400. kemudian amplas bagian yang sudah di dempul. Jika pengamplasan sudah halus dan merata, lakukan pencucian dengan air sabun lalu gosokan dengan amplas yang ukuranya 1000. Gosokkan berulang-ulang hingga halus agar mendapatkan hasil yang baik. Setelah pencucian dengan air sabun colek selesai, bersikan dengan kanebo atau majun yang kering.Kemudian dilanjutkan dengan pembuatan platform berdasarkan cetakan yang telah dibuat.

3. TahapanTahapan Pembuatan Cetakan *Minus*:

- Pemberian Wax

Pemberian wax pada tiap-tiap bagian cetakan *minus* kapal yang akan digunakan untuk mencetak semua bagian kapal. Pemberian wax ini dengan dengan cara digosok dengan menggunakan majun atau kain sampai merata dan sampai

terlihat licin. Tujuannya agar pada saat badan kapal dilepas tidak terlalu melekat dan kering dengan cetakan sehingga hasil cetakan menjadi halus.

- Pemberian *Gell Coat*

Lapisan *Gell Coat* yaitu campuran resin dengan HDK/Sylica dan pigmen (pewarna). *Gell Coat* digunakan sebagai lapisan kedua setelah wax, fungsinya agar warna lebih mengkilap dan tidak berubah selamanya. Selain itu, *Gell Coat* berfungsi untuk mengisi celah-celah sempit seperti ujung depan dan bawah kapal yang susah dijangkau oleh mat.

Gambar IV.6 Hasil setelah diberi *Wax* dan *Gell Coat*

- Pelapisan dengan bahan fiber

Pelapisan semua bagian cetakan dengan bahan fiber yang terdiri dari mat , resin ,dan katalis. Mat dibentangkan hingga menutup seluruh bagian catakan *minus* lambung kapal, kemudian diolesi dengan resin yang sudah tercampur dengan katalis. Kemudian resin diratakan dan ditekan-tekan menggunakan kuas agar resin bisa menyebar rata di seluruh permukaan mat. Sebelum kering, proses pelapisan ini diulang lagi untuk menambah lapisan bahan fiber hingga lapisan fiber mencapai 2 lapis. Hal ini dilakukan agar lambung kapal kuat dan tidak mudah pecah ketika pemisahan dengan catakan *minus* lambung kapal.

Gambar IV.7 Proses Pelapisan dengan Bahan Fiber

Gambar IV.8 Hasil Pelapisan dengan Bahan Fiber

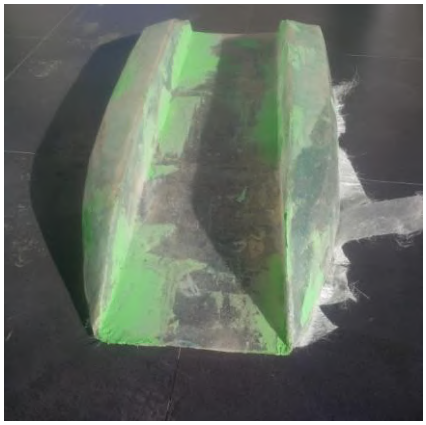
- Pemisahan antara cetakan *minus* dengan lambung kapal

Setelah resin sudah benar-benar kering, dilakukan pemisahan antara cetakan *minus* dengan lambung kapal. Pemisahan ini dilakukan dengan cara mengisi air pada sela-sela antar kedua lapisan, kemudian ditiup angin dengan kompresor bertekanan tinggi. Setelah kedua bagian terpisah, kemudian diambil bagian lambung kapal yang sudah jadi lalu dibersihkan.

Gambar IV.9 Pemisahan Antara Cetakan Minus dengan Lambung Kapal

- Penghalusan lambung kapal

Setelah lambung kapal bersih, kemudian dilanjutkan proses pendempulan untuk melapisi celah-celah cetakan dan untuk meratakan cetakan agar saat digunakan menjadi rata dan tidak memiliki celah. Setelah kering, proses dilanjutkan dengan mengampas lambung kapal. Pengamplasan adalah proses penghalusan bagian yang sudah didempul untuk mendapatkan hasil yang baik. Pertama siapkan amplas dengan ukuran 400. kemudian amplas bagian yang sudah di dempul. Jika pengamplasan sudah halus dan merata, lakukan pencucian dengan air sabun lalu gosokan dengan amplas ukuran 1000. Gosokan berulang-ulang hingga halus agar mendapatkan hasil yang baik. Setelah pencucian dengan air sabun colek selesai, bersikan dengan kanebo atau majun yang kering.



Gambar IV.10 Penghalusan Lambung Kapal

- Pemberian warna dasar

Pemberian warna dasar abu-abu dilakukan pada lambung kapal yang sudah halus. Pemberian warna pada lambung kapal bertujuan untuk mencegah cat mudah terkelupas ketika proses pengecatan nanti.

Gambar IV.11 Pemberian Warna Dasar

2. **Proses Pembuatan Dek dan *Main Mast***

Tahap kedua adalah pembuatan bagian dek dan *main mast*. Tahapan-tahapannya yaitu:Pengukuran Bentuk Dek, Palkah, dan *Main Mast.* Pengukuran dilakukan secara manual berdasarkan desain yang telah dibuat. Setelah mendapatkan ukuran skalanya, maka dibuat gambar bukaan kulit untuk bagian-bagian ini secara manual. Dek, palkah, dan *main mast* dibuat menggunakan bahan PVC foam 5 mm. Sementara tutup palkah dibuat dengan menggunakan bahan PVC foam 3 mm.

Gambar IV.12 Pengukuran Bentuk Dek, Palkah, dan *Main Mast*

1. Penyusunan Dek dan Palkah

   Setelah pengukuran dek selesai, kemudian desain dipotong mengikuti bentuknya. Setelah semua bagian sudah dipotong, lalu dilakukan penyusunan bagian-bagian tersebut.

Gambar IV.13 Penyusunan Dek dan Palkah

2. Penyambungan Lambung dengan Dek

   Langkah selanjutnya yaitu menyaSemua bagian pada kapal dilapisi dengan menggunakan bahan fiber yang terdiri dari mat, ropping, katalis dan resin pelapisan

ini di gunakan pada saat semua cetakan sudah jadi karena pelapisan dibuat juga untuk mempersatukan semua cetakan agar menjadi satu. Pelapisan ini juga bertujuan untuk mempekuat hasil cetakan dengan cara digabungkan dan dilapisi lagi.

Gambar IV.14 Penyusunan dan Penyambungan Dek

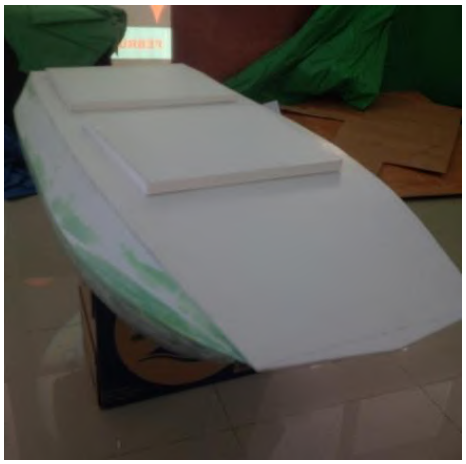
2. Penghalusan Badan Kapal Secara Keseluruhan

Pada sela-sela kapal yang masih berongga. Dilakukan pemenutupan bagian-bagian ini diperlukan tali kecil (mahjong). Cara memasangnya adalah dengan cara memasukkan tali tersebut kedalam sela-sela kecil hingga tertutup semua dan tidak ada rongga lagi. Lalu dilanjutkan dengan meratakan badan dengan dempul yang bahannya terdiri dari: dempul, katalis, dan kepi. Ambil dempul dan katalis secukupnya lalu di aduk dengan kepi hingga merata,setelah bahan dempul sudah siap lakukan pendempulan pada: dek kapal, dinding kapal, lantai kapal,.pintu kapal, dan tutup palka. Lakukan tiga kali pendempulan hingga merata.

Kemudian sambungan antara lambung dengan dek diamplas hingga permukaannya terlihat rapi dan mulus. Perapian ini secara bertahap dari depan kebelakang. Kemudian dilakukan lagi pengamplasan pada bagian keseluruhan kapal bertahap dari depan ke belakang dan dari bawah sampai ke atas. Hal ini dilakukan secara berulang-ulang agar mendapat kualitas yang baik.

Setelah lambung kapal bersih, kemudian dilanjutkan proses pendempulan untuk melapisi celah-celah cetakan dan untuk meratakan cetakan agar saat digunakan menjadi rata dan tidak memiliki celah. Setelah kering, proses dilanjutkan dengan mengamplas lambung kapal. Pengamplasan adalah proses penghalusan bagian yang sudah didempul untuk mendapatkan hasil yang baik. Pertama amplas bagian yang sudah di dempul dengan amplas 400. Jika pengamplasan sudah

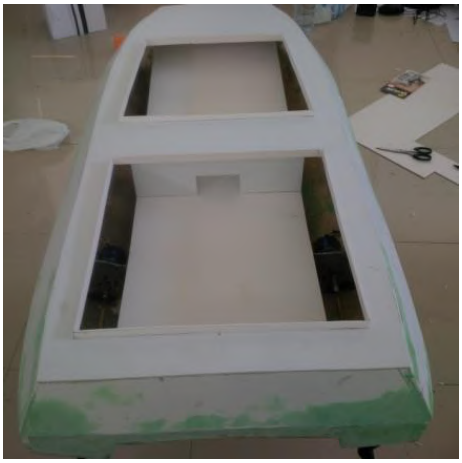
halus dan merata, lakukan pencucian dengan air sabun lalu gosok lagi dengan amplas yang ukuranya 1000. Gosokan berulang-ulang hingga halus agar mendapatkan hasil yang baik. Lalu lakukan pencucian dengan air dan bersikan dengan kanebo atau majun yang kering.



Gambar IV.15 Penghalusan Badan Kapal Secara Keseluruhan

3. Pemasangan Sekat untuk Ruangan

Pemasangan sekat di dalam kapal menggunakan lembaran PVC *foam* yang diberi penguat. Pemasangan sekat ini dipasang pada ruang-ruang di dalam purwarupa. Di bagian ruang belakang untuk komponen sistem propulsi dan di bagian ruang depan untuk komponen sistem autopilot. Pemasangan sekat bertujuan untuk memberi batas pada ruangan di dalam kapal agar jika terjadi kecelakaan di satu ruangan tidak cepat menyebar ke ruangan lain. Pemasangan sekat di pinggir dilapisi dengan bahan fiber agar menjadi kuat dan kedap air. Pemasangan sekat ini dilakukan pada saat lambung sudah selesai.



Gambar IV.16 Pemasangan Sekat untuk Ruangan

4. Pemberian Warna Dasar

Pemberian warna dasar abu-abu dilakukan pada lambung kapal yang sudah halus. Pemberian warna pada lambung kapal bertujuan untuk mencegah cat mudah terkelupas ketika proses pengecatan nanti.

Gambar IV.17 Pemberian Warna Dasar

5. Pengecatan kapal

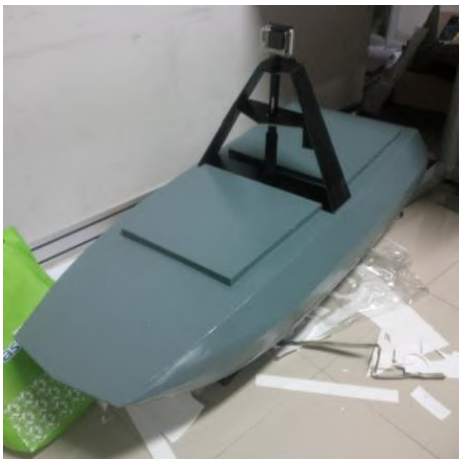
Pengecatan adalah proses pelapisan dengan menggunakan cat warna. Sebelum pengecatan dilakukan, badan kapal dibersihkan terlebih dahulu dengan menggunakan air. Setelah semua sudah bersih, pengecatan siap di lakukan. Pengecatan dilakukan sampai 2 kali untuk membuat warna lebih tebal. Sesudah pengecatan selesai, lalu dilapisi dengan anti gores (*clear)* agar cat kapal tidak mudah memudar atau luntur.

Gambar IV.18 Pemberian Warna Kapal

2. Pemasangan dan *finishingMain Mast*

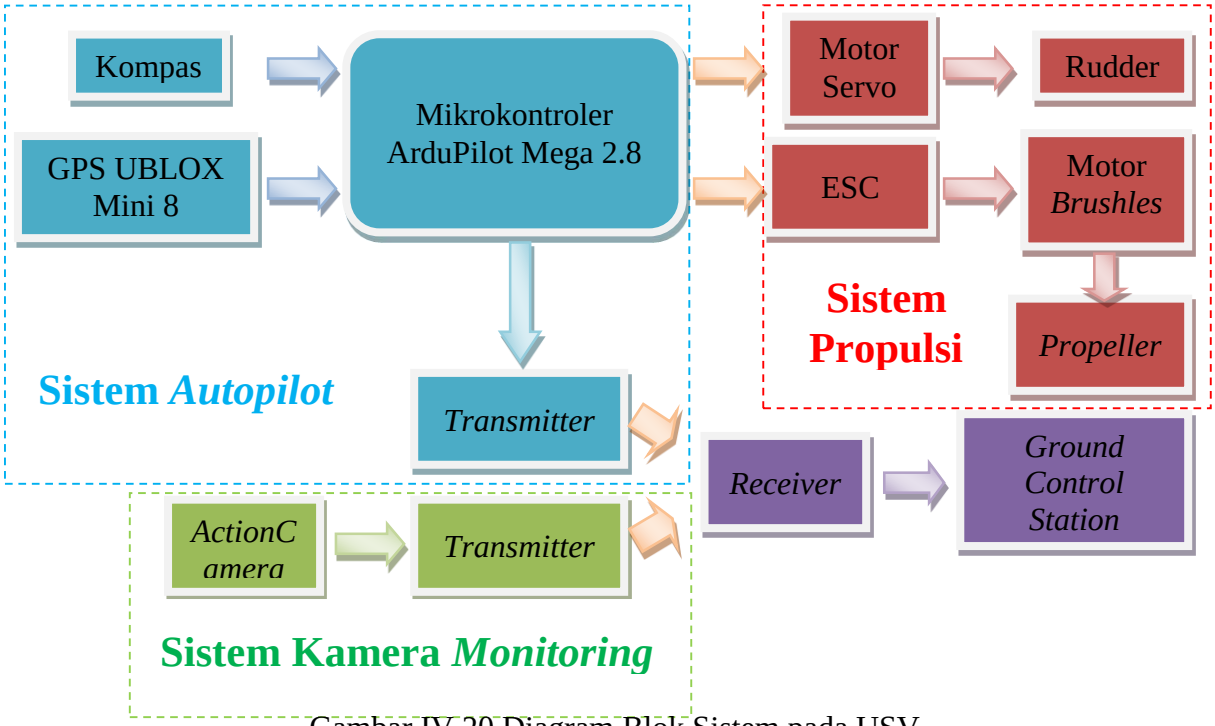
Main mast dibuat menggunakan lembaran PVC *foam* 5 mmyang diberi penegar dan *bracket*. Pemasangan *main mast* ini dipasang pada bagian tengah

purwarupa USV. Pemasangan *main mast* bertujuan untuk peletakan komponen sistem kamera dan sistem navigasi modul GPS purwarupa USV. Pemasangan *main mast* ini dilakukan pada saat lambung sudah selesai. *Main mast* harus dipasang secara tegak lurus dan harus kokoh karena pada saat purwarupa USV bergerak, posisi GPS dan kamera harus diusahakan sebisa mungkin dalam kondisi stabil sehingga tidak mengganggu navigasi dan proses *monitoring* kapal.



Gambar IV.19 Pemasangan dan Finishing *Main Mast*

IV.3. Diagram Blok Sistem



Gambar IV.20 Diagram Blok Sistem pada USV

Berdasarkan diagram diatas, sistem pada USV dibagi menjadi 3 bagian. Bagian pertama adalah sistem propulsi, yaitu sistem yang mengatur penggerak dan kemudi kapal. Bagian kedua adalah sistem autopilot, yaitu sistem yang mengatur navigasi, mode manual dan mode autopilot pada USV. Bagian ketiga adalah sistem kamera*monitoring*, yaitu sistem yang mengatur kamera *monitoring* pada USV.

Agar ketiga sistem ini dapat berjalan dan berkomunikasi secara optimal dibutuhkan beberapa komponen pada USV seperti dibawah ini:

1. Kompas, berfungsi sebagai penunjuk arah mata angin pada USV,sehingga memudahkan dalam mengendalikan arah manuver USV.
2. GPS UBLOX M8N, berfungsi sebagai *receiver* data *Global Positioning System* (GPS) dari satelit yang mengorbit diatas permukaan bumi. Dengan adanya GPS, posisi koordinat USV dapat diketahui.
3. ArduPilot Mega 2.8, mikrokontroler yang berfungsi sebagai pusat pengendali, pengolah, dan pemroses data maupun sinyal pada USV.
4. Motor servo, merupakan aktuator yang akan menggerakkan *rudder* (kemudi) sehingga USV dapat bermanuver berhalauan ke kiri maupun ke kanan dengan baik.
5. *Rudder,* sebagai alat yang digunakan untuk mengubah dan menentukan arah gerak kapal, baik arah lurus maupun belok kapal.
6. *Electronic Speed Control*(ESC),) yang berfungsi sebagai pengatur kecepatan motor, selain itu juga berfungsi untuk menaikan-turunkan jumlah arus yang diperlukan oleh motor.
7. Motor *brushless,* merupakan aktuator yang berfungsi untuk memberikan gaya dorong *(thruster)*pada *propeller*sehingga USV dapat bergerak maju ataupun mundur.
8. *Propeller*, untuk mengubah gaya rotasi dari mesin menjadi gaya propulsif sebagai gaya dorong (*Thrust*) untuk kapal
9. *Transmitter* dan *Receiver*, berfungsi untuk komunikasi antara *Ground Control Station* (GCS) ke USV ataupun sebaliknya, media komunikasi ini menggunakan media udara. Dengan adanya modul ini memungkinkan data pada USV dapat dipantau melalui GCS.
10. Kamera*Action*, sebagai alat untuk fungsi *monitoring* pada USV.
11. Laptop dan Tablet , berfungsi memantau data dari *Ground Control Station* (GCS).

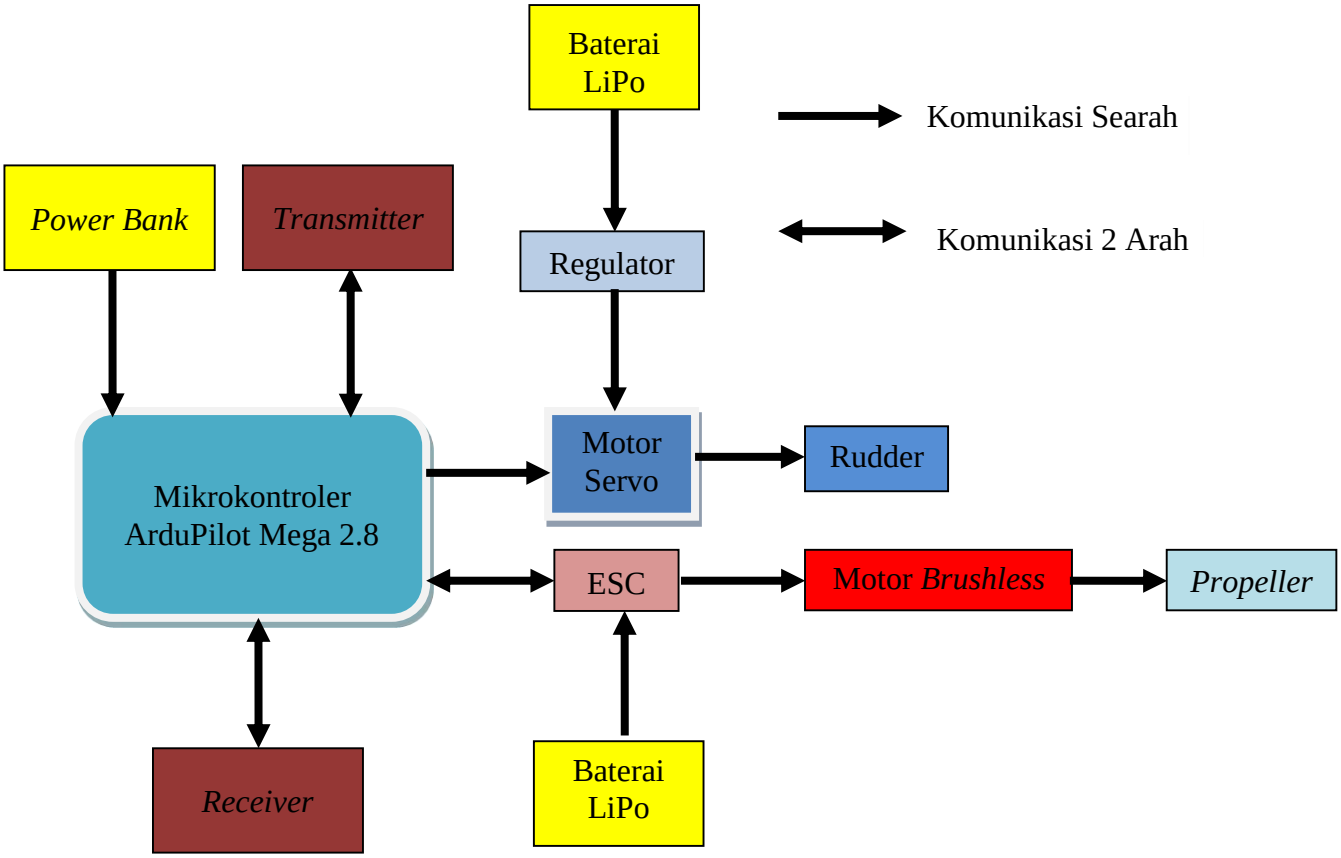
Adapun penjelasan dari diagram sistem pada gambar diatas akan dibahas lebih lanjut. pada sub bab seperti desain dan pembuatan sistem propulsi, sistemautopilot, dan sistem kamera*monitoring*.

IV.4. Sistem Propulsi

Sistem propulsi adalah bagian dari sistem yang memungkinkan agar USV dapat bergerak dan bermanuver dengan baik. Sistem propulsi USV ini menggunakan alat penggerak berupa Motor DC *Brushless*, *Electronic Speed Control*(ESC), Baterai LiPo, Regulator, *Transmitter*dan *Receiver*, Motor Servo, *Rudder*, dan *Propeller* (baling-baling). Dari semua komponen tersebut, pengaturan sistem dilakukan oleh mikrokontroler ArduPilot Mega 2.8 sebagai pusat pengendali USV yang diakses dan dikalibrasi terlebih dahulu oleh pilot di *Ground Control Station* baiksecara langsung menggunakan kabel USB maupun secara tidak langsung menggunakan komunikasi *wireless.*

IV.4.1. Desain Sistem Propulsi

Berikut adalah diagram desain sistem propulsi untuk purwarupa tes model *Unmanned Surface Vehicle* yang ditunjukkan pada Gambar IV.2.



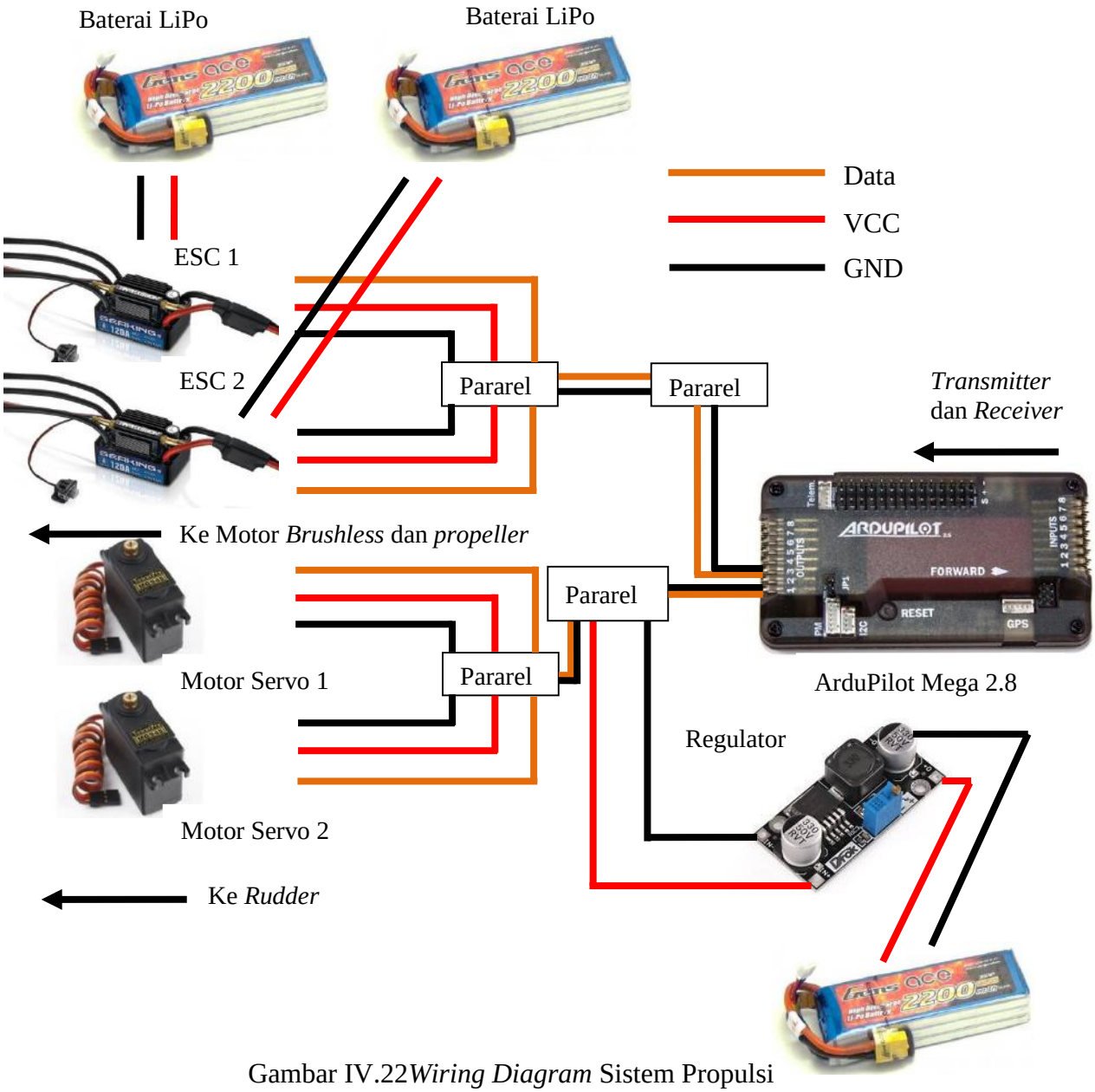
Gambar IV.21 Diagram Sistem Propulsi

Keterangan untuk diagram :

1. Baterai Li-Po Tiger 3S 12 Volt 2200 mAh, sebagai catu daya utama untuk komponen propulsi. Baterai yang digunakan adalah sebanyak 3 buah dengan rincian 1 buah untuk menggerakkan motor servo dan 2 buah untuk motor *Brushless. Output* tegangan yang dihasikan adalah sebesar 12 volt.
2. Regulator, untuk mengurangi tegangan dari baterai ketika dialirkan ke motor servo. *Output*tegangan 12 volt baterai terlalu besar untuk motor servo yang hanya memiliki tegangan 6 volt sehingga harus diregulasi. Regulator memiliki *switch* yang dapat diatur untuk mendapatkan *output*tegangan yang diinginkan, tetapi *input* tegangan yang dimiliki adalah konstan 12 volt.
3. Motor Servo Tower Pro MG 945, sebagai *steering gear* untuk menggerakkan *rudder. Input*tegangan yang dimiliki adalah 6 volt. Motor servo ini memiliki kekuatan untuk menggerakkan benda dengan berat sebesar 10 kilogram.
4. *Suction Water Rudder,* kemudi kapal merupakan suatu alat kapal yang digunakan untuk mengubah dan menentukan arah gerak kapal, baik arah lurus maupun belok kapal.*Rudder* memiliki tinggi 15 cm dengan lebar 3.5 cm.
5. *Electronic Speed Control*(ESC)Hobby King 90A Boat ESC 4A SBEC, sebagai pengatur kecepatan motor. Selain itu juga berfungsi untuk menaikan jumlah arus yang diperlukan oleh motor, karena arus yang dibutuhkan berbeda-beda tergantung kecepatan kapal. Arus yang dimiliki oleh ESC ini adalah maksimal 90 Ampere ketika kapal dalam kecepatan maksimal dan minimal 0 Ampere ketika kapal dalam keadaan berhenti.
6. Motor *Brushless* DC Turnigy Aquastar 1460 KV, sebagai motor utama untuk menggerakkan kapal. Arus maksimal yang dapat diterima oleh motor ini adalah 90 Ampere. Putaran yang dihasilkan oleh motor ini mencapai 1460 rpm/volt.
7. *Receiver* RadioLink R9DS2.4 Hz, berfungsi untuk komunikasi antara *Ground Control Station* (GCS) ke USV ataupun sebaliknya secara *wireless*. Dengan adanya modul ini memungkinkan data pada USV dapat dipantau melalui GCS.
8. *Propeller* 2 *blades* 50 mm, sebagai pendorong kapal.
9. Mikrokontroler ArduPilot Mega 2.8, sebagai pusat pengendali USV yang diakses dan dikalibrasi terlebih dahulu oleh pilot di *Ground Control Station* (GCS) baiksecara langsung menggunakan kabel USB maupun secara tidak langsung menggunakan komunikasi *wireless.* Mikrokontroler ArduPilot Mega 2.8 akan diakses melalui *software Mission Planner* oleh pilot melalui laptop di *Ground Control Station* (GCS).

10. *Transmitter* Radio Telemetry 915 MHz yang berfungsi untuk menghubungkan ArduPilot Mega 2.8 pada USV dengan *Mission Planner* di laptop pada *Ground Control Station* (GCS). Dengan adanya modul ini memungkinkan data pada USV dapat dipantau melalui GCS.
11. *Power Bank* Hippo 11000 mAh, sebagai catu daya utama untuk Mikrokontroler ArduPilot Mega 2.8. *Output* tegangan yang dihasikan adalah sebesar 5 volt. Untuk mensuplai Mikrokontroler ArduPilot Mega 2.8, arus maksimal yang diberikan adalah 1 Ampere.

Untuk *wiring diagram* antara *Electronic Speed Control* (ESC) dan Motor servo dengan ArduPilot Mega 2.8 adalah sebagai berikut :



Gambar IV.22*Wiring Diagram* Sistem Propulsi

Berdasarkan *wiring diagram*, terdapat beberapa garis-garis lurus yang berbeda warna yang menggambarkan jenis kabel-kabel yang ada pada sistem propulsi purwarupa USV. Selain itu, pada ArduPilot Mega 2.8 terdapat beberapa pin dari pin 1-8 pada *input* dan *output.* Untuk *output* pada pin 1 adalah untuk motor servo. Sementara *output* pada pin 3 adalah untuk ESC. Fungsi penggunaan *output* ini dapat diketahui dari *data sheet* ArduPilot Mega 2.8.Untuk pin selain pin 1 & 3 tidak digunakan karena tidak mendukung fungsi purwarupa USV.

Untuk penjelasan garis-garis pada *wiring diagram* adalah sebagai berikut. Garis oranye adalah kabel data yang berisi perintah yang diberikan oleh ArduPilot Mega 2.8 berdasarkan pengaturan oleh pilot di *Ground Control Station* (GCS). Data yang berisi perintah dari ArduPilot Mega 2.8 diteruskan ke ESC dan motor servo. ESC akan meneruskan perintah untuk mengatur kecepatan motor *Brushless,* sementara motor servo akan menerima perintah untuk mengatur pergerakan *rudder.*

Garis merah adalah VCC atau voltase. Garis ini adalah kebel yang berisi tegangan voltase yang dialirkan dari Baterai Li-Po Tiger 3S 12 Volt 2200 mAh sebagai catu daya utama untuk komponen propulsi. Baterai yang digunakan adalah sebanyak 3 buah dengan rincian 1 buahuntuk menggerakkan kedua motor servo dan 2 buah untuk motor *Brushless. Output* tegangan yang dihasikan adalah sebesar 12 volt. Namun, untuk motor servo, *input* tegangan yang dapat diterima adalah 5 volt sehingga *output* dari baterai 12 volt akan di regulasi menjadi 5 volt, setelah itu baru dialirkan ke motor servo.

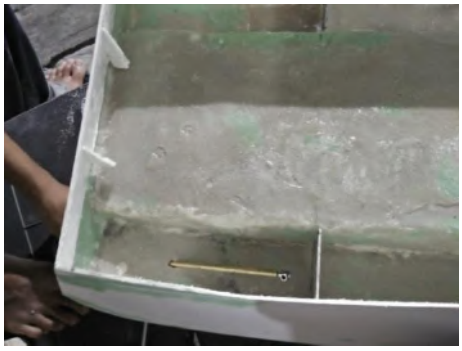
Garis hitam adalah *ground*. Garis ini adalah kabel yang berisi voltase 0 volt dan sebagai bagian *minus.* Fungsi dari kebel *ground* adalah sebagai proteksi peralatan elektronik sehingga dapat mencegah kerusakan akibat daya bocor tegangan dan menetralisir *noise* yang disebabkan kualitas komponen yang tidak standar.

IV.4.2. Proses Pembuatan Sistem Propulsi

Berikut adalah proses pembuatan sistem propulsi untuk purwarupa tes model *Unmanned Surface Vehicle.*

1. Pemasangan *Tunnel Shaft*

   Langkah-langkahnya adalah mengatur kemiringan antara sudut memanjang kapal dengan kemiringan *shaft*. Sudut maksimal kemiringan adalah 5^{o}. Setelah didapatkan kemiringannya, lalu bagian *stern*kapal dibor sesuai ukuran diameter *tunnel shaft.*



Gambar IV.23 Pemasangan *Tunnel Shaft*

2. Pemasangan Motor *Brushless*

Langkah pertama adalah membuat fondasi mesin dengan mengukur posisi peletakan mesin menyesuaikan sudut kemiringan *tunnel shaft.* Setelah itu pasang fondasi mesinnya. Langkah kedua adalah, bor fondasi kapal untuk memasang pembautan pangkon mesin, lalu pasanga mesindan pasang mur untuk memperkuat mesin agar tidak goyang. Masukkan *shaft* lewat buritan kapal, lalu hubungkan dengan mesin menggunakan *joint* sesuai ukuran diameter *shaft* dan *motor shaft.*

Gambar IV.24 Pemasangan Shaft dan Motor *Brushless*

3. Pemasangan *Propeller*

Pada bagian ujung *shaft*, *propeller*dipasang pada poros dan menentukan titik seimbangnya. Laludikuatkan dengan menggunakan baut.

Gambar IV.25 Hasil Pemasangan *Propeller*

4. Pemasangan *Rudder*

Langkah-langkahnya adalah mengukur peletekkan *rudder* dengan memastikan bahwa *blade* sudah tercelup semua. Kemudian melubangi *transom* dan pasang dengan baut dengan sempurna agar tidak tergeser.

Gambar IV.26 Hasil Pemasangan *Rudder*

5. Pemasangan Motor Servo

Langkah-langkahnya adalah mengukur ketinggian poros penghubung *rudder* dengan motor servo untuk memastikan bahwa poros penghubung sudah lurus antara *rudder* dengan motor servo. Kemudian memasang dudukan motor servo dan dilanjutkan memasang motor servonya. Setalah itu, motor servo dibaut pada dudukan servo dan dipastikan bahwa posisi motor servo tidak akan berubah posisi ketika mulai bekerja. Lalu pasang ruji untuk menghubungkan motor servo dengan *rudder* di bagian luar belakang purwarupa. Lubangi bagian transom purwarupa agar ruji bisa lewat. Pada bagian yang lubang, dipasang *rubber* untuk mencegah air masuk ketika purwarupa bergerak di air.

6. PemasanganESC

Pemasangan ESC diusahakan sedekat mungkin dengan Motor *Brushless* untuk menghemat panjang kabel. Pada bagian bawah ESC, direkatkan dengan menggunakan *double tape* ke bagian *bridge* purwarupa agar tidak bergeser ketika purwarupa mulai bergerak.

Gambar IV.27 Pemasangan ESC

7. Pemasangan Kabel

   Pemasangan kabel dilakukan sesuai desain *wiring diagram*.

8. Pemasangan Regulator

   Pemasangan Regulator dilakukan sesuai desain *wiring diagram*. Peletakannya diusahakan sedekat mungkin dengan baterai untuk menghemat panjang kabel.

9. Pemasangan Baterai

   Peletakkan baterai diusahakan sedekat mungkin dengan Motor *Brushless* untuk menghemat panjang kabel. Baterai diletakkan di dalam kotak kedap air yang sudah direkatkan ke bagian *bridge* purwarupa agar tidak bergeser ketika purwarupa mulai bergerak. Selain itu, baterai juga digunakan sebagai *ballast* pada purwarupa.

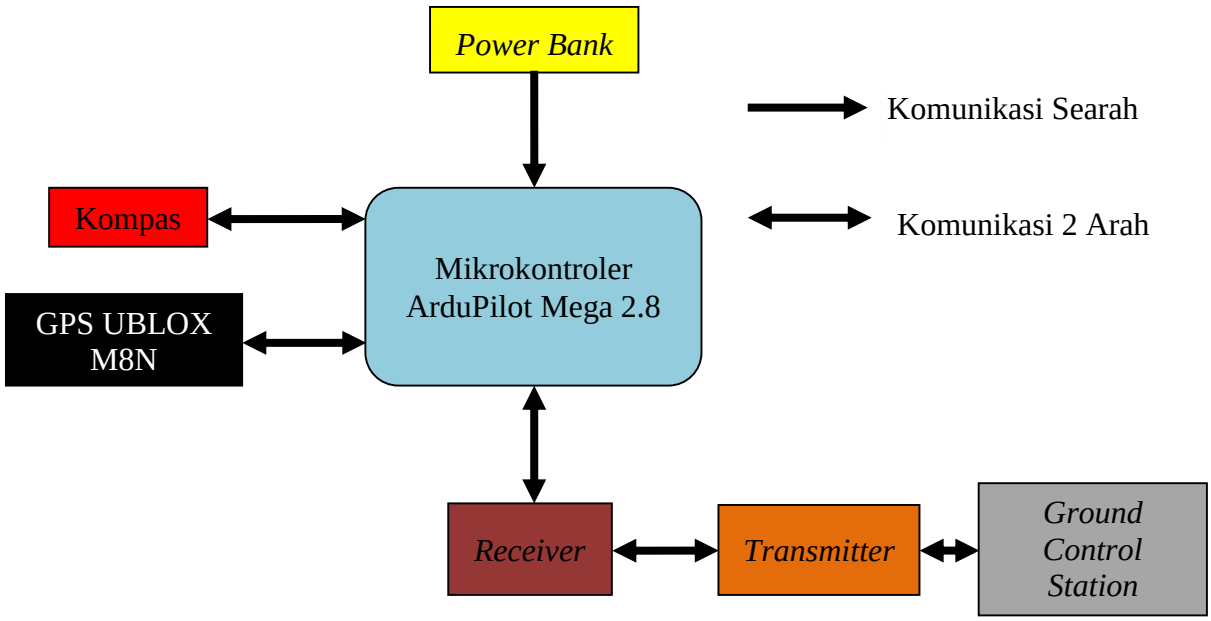
Gambar IV.28 Hasil Instalasi Sistem Propulsi

IV.5. Sistem Autopilot

Sistem autopilotadalah bagian dari sistem yangmemungkinkan agar USV dapat bergerak dan bermanuver sesuai dengan navigasi GPS dan kompas. Sistem autopilot USV ini didukung komponen berupaMikrokontroler ArduPilot Mega 2.8, *Mission Planner*, Modul GPS, Kompas,*Transmitter* dan *Receiver*,dan *power bank*. Dari semua komponen tersebut, pengaturan sistem dilakukan oleh mikrokontroler ArduPilot Mega 2.8 sebagai pusat pengendali USV yang diakses dan dikalibrasi terlebih dahulu oleh pilot di *Ground Control Station* baik secara langsung menggunakan kabel USB maupun secara tidak langsung menggunakan komunikasi *wireless.*

IV.5.1. Desain Sistem Autopilot

Berikut adalah diagram desain sistem autopilot untuk purwarupa tes model *Unmanned Surface Vehicle* yang ditunjukkan pada Gambar IV.29.



Gambar IV.29 Diagram Sistem Autopilot

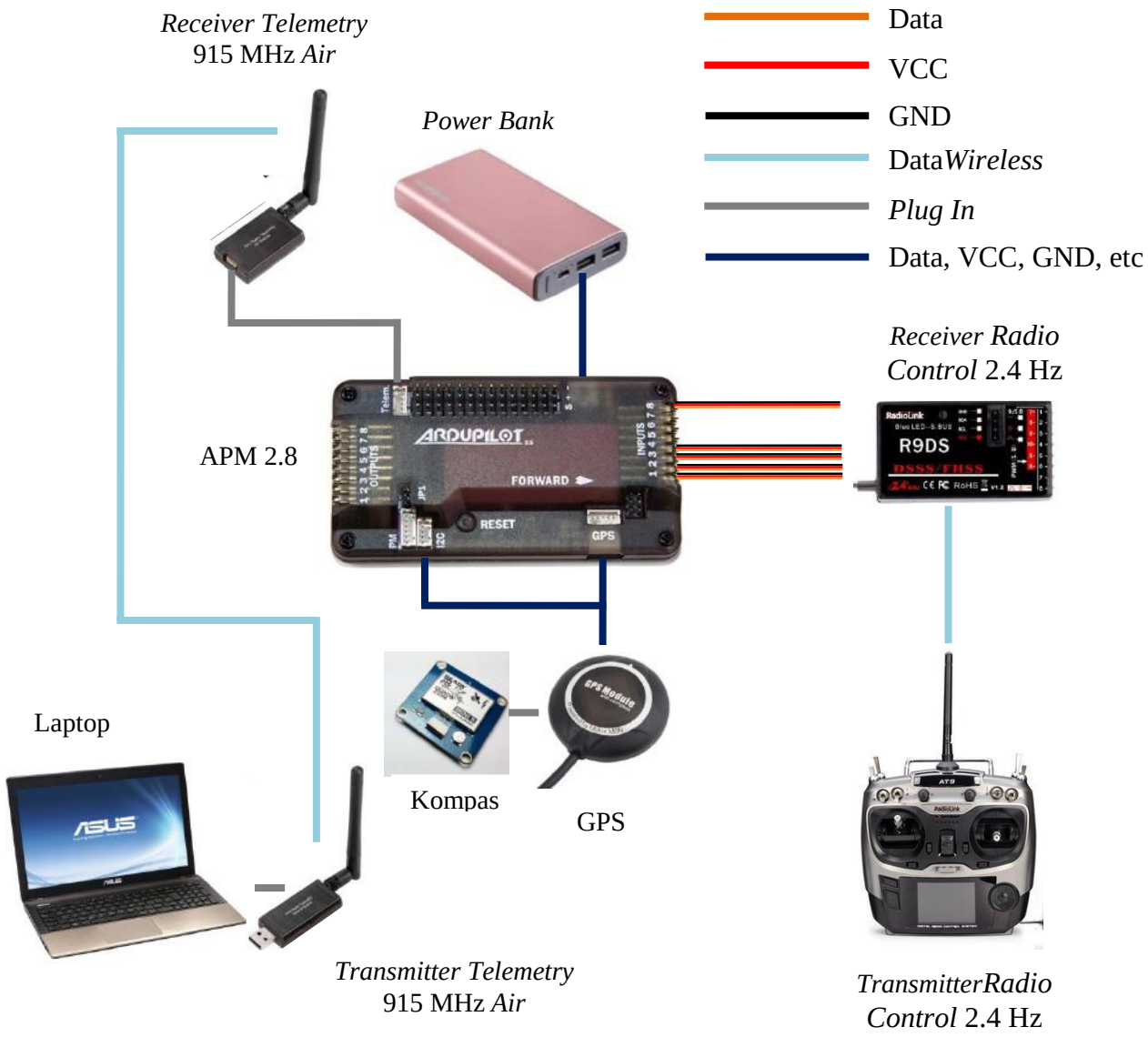
Keterangan untuk diagram :

1. *Power Bank* 11000 mAh, Sebagai catu daya utama untuk komponen sistem autopilot. *Output* tegangan yang dihasilkan adalah sebesar 5 Volt dengan arus sebesar 1 Ampere.
2. Mikrokontroler ArduPilot Mega 2.8, sebagai komponen utama sistem autopilot. Berfungsi sebagai kendali navigasi secara manual, autopilot dan olah gerak pada kapal. Sensor yang digunakan adalah GPS dan kompas. Penempatan GPS dan kompas ini bisa

diletakkan pada *casing* atau wadah ArduPilot itu sendiri ataupun tempat yang lainnya. ArduPilot Mega 2.8 diakses dan dikalibrasi terlebih dahulu oleh pilot di *Ground Control Station* (GCS) baik secara langsung menggunakan kabel USB maupun secara tidak langsung menggunakan komunikasi *wireless* . Mikrokontroler ArduPilot Mega 2.8 akan diakses melalui *software Mission Planner* oleh pilot melalui laptop di *Ground Control Station* (GCS). Setelah itu, dikalibrasi untuk menentukan keakuratan koordinat GPS, keakuratan arah kompas, kecepatan *Electronic Speed Control* (ESC) dan Motor *Brushless,* dan pergerakan Motor Servo serta *rudder.* Ketika proses kalibrasi sudah selesai, maka dimasukkan koordinat-koordinat GPS untuk melakukan misi. Pergerakan purwarupa USV dilakukan berdasarkan perintah yang diberikan dan diolah oleh ArduPilot Mega 2.8.APM 2.8 harus diletakkan menghadap arah depan purwarupa USV agar dapat bekerja.

3. Kompas, untuk memberikan arah tujuan USV, sehingga sangat membantu dalam bidang navigasi purwarupa USV. Kompas harus diletakkan menghadap arah depan purwarupa USV agar dapat bekerja.
4. GPS UBLOX M8N, untuk memberikan posisi dan kecepatan tiga-dimensi serta informasi tentang posisi, kecepatan, percepatan ataupun waktu pada USV. GPS harus diletakkan pada *main mast* purwarupa untuk dapat bekerja menangkap sinyal dari satelit.
5. *Transmitter*dan *Receiver,* berfungsi untuk komunikasi antara *Ground Control Station* (GCS) ke USV ataupun sebaliknya, media komunikasi bersifat tanpa kabel *(wireless).* Pada USV ini digunakan 2 *Transmitter* dan *Receiver.* Yang pertama adalah *Radio Control* 2.4 Hz untuk mengendalikan USV secara manual dan memiliki *switch* untuk berpindah ke mode autopilot. Yang kedua adalah *Radio Telemetry* 915 MHz*Air* untuk menampilkan dan menginformasikan mengenai GPS dan kompas dari USV ke *Ground Control Station* (GCS).
6. *Ground Control Station*(GCS), sebagai pusat pengendalian USV oleh pilot. Perangkat yang digunakan adalah *Radio Control*2.4 Hz dan Laptop. *Radio Control*2.4 Hz untuk mengendalikan USV secara manual dan memiliki *switch* untuk berpindah ke mode autopilot. Laptop digunakan untuk menampilkan dan menginformasikan mengenai GPS dan kompas dari USV dengan memasang *Telemetry* 915 MHz*Ground.*

Untuk *wiring diagram* antara ArduPilot Mega 2.8 dengan GPS, kompas, *transmitter* dan receiver adalah sebagai berikut :



Gambar IV.30*Wiring Diagram* Sistem Autopilot

Berdasarkan *wiring diagram*, terdapat beberapa garis-garis lurus yang berbeda warna yang menggambarkan jenis kabel-kabel yang ada pada sistem propulsi purwarupa USV. Selain itu, pada ArduPilot Mega 2.8 terdapat beberapa pin dari pin 1-8 pada *input* dan *output.* Untuk *input* pada pin 1 adalah untuk motor servo,*input*pada pin 2 adalah untuk GPS,*input* pada pin 3 adalah untuk ESC,*input* pada pin 4 adalah untuk kompas, dan *input* pada pin 8 adalah untuk mode manual. Fungsi penggunaan *input* ini dapat diketahui dari *data sheet* ArduPilot Mega 2.8.Untuk pin 5-7 tidak digunakan karena tidak mendukung fungsi purwarupa USV. Lalu ada juga pin untuk USB yang dihubungan dengan *power bank* sebagai *power* untuk ArduPilot Mega 2.8, pin GPS serta kompas, serta pin untuk *Radio Telemetry* 915 MHz*Air.*

Untuk penjelasan garis-garis pada *wiring diagram* adalah sebagai berikut. Garis oranye adalah kabel data yang berisi perintah yang saling diinformasikan oleh ArduPilot Mega 2.8 dan *Radio Control* 2.4 Hzberdasarkan pengaturan oleh pilot di *Ground Control Station* (GCS). Garis merah adalah VCC atau voltase. Garis ini adalah kebel yang berisi tegangan voltase yang dialirkan dari *power bank* sebagai catu daya utama untuk komponen autopilot. *Output* tegangan yang dihasikan adalah sebesar 5 volt. Garis merah adalah *ground*. Garis ini adalah kabel yang berisi voltase 0 volt dan sebagai bagian *minus.* Fungsi dari kebel *ground* adalah sebagai proteksi peralatan elektronik sehingga dapat mencegah kerusakan akibat daya bocor tegangan dan menetralisir *noise* yang disebabkan kualitas komponen yang tidak standar.

Untuk garis biru muda adalah data komunikasi yang dipancarkan dan diterima secara *wireless* oleh *Radio Control* 2.4 Hz dan *Radio Telemetry* 915 MHzdari USV ke *Ground Control Station* (GCS).Untuk garis abu-abu adalah tanda *plug in*, artinya perangkat saling tersambung secara langsung dengan perangkat lainnya. Untuk garis biru tua, adalah kabel besar yang berisi kabel-kabel kecil untuk Data, VCC, GND, dan kabel lainnya.

IV.5.2. Pembuatan Sistem Autopilot

Berikut adalah proses pembuatan sistem autonomous untuk purwarupa tes model *Unmanned Surface Vehicle.*

1. Pemasangan Mikrokontroler ArduPilot Mega 2.8

   Mikrokontroler ArduPilot Mega 2.8 diletakkan pada wadah di bagian ruangan depan purwarupa. Posisi peletaknnya harus lurus sejajar dengan permukaan bumi dan menghadap ke depan purwarupa. Hal ini untuk mencegah terjadinya *drifting* pada komponen kompas karena kompas dipasang di dalam APM2.8.

2. Pemasangan Perangkat GPS dan Kompas

   GPS diletakkan pada wadah di bagian atas *main mast* purwarupaPosisi peletaknnya harus lurus sejajar dengan permukaan bumi dan menghadap ke depan purwarupa. Hal ini untuk mencegah terjadinya *error* pada komponen GPS. Sementara kompas diletakkan pada bagian dalam APM 2.8. Posisi peletaknnya harus lurus sejajar dengan permukaan bumi dan menghadap ke depan purwarupa. Hal ini untuk mencegah terjadinya *drifting* pada komponen kompas.

3. Pemasangan Kabel

   Pemasangan kabel dilakukan sesuai desain *wiring diagram.*

Gambar IV.31 Saluran Kabel Penghubung

4. Pemasangan Perangkat *Transmitter* dan *Receiver*

Pemasangan perangkat *transmitter* dan *receiver* dilakukan sesuai desain *wiring diagram.*

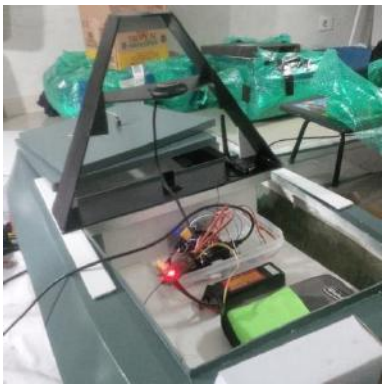
5. Menghubungkan Tiap Perangkat Antar Pin

Berikut adalah *input* dan *output* pin tiap komponen yang ditunjukkan pada TabelIV.1.

Tabel IV.1 *Input* dan *Output P*in Tiap Komponen

No	**Komponen/Modul**	***Input* APM 2.8**	***Output*APM 2.8**	**Keterangan**
1	GPS UBLOX Mini 8	Pin GPS	Pin 2	Ke APM 2.8, lalu ke *Receiver RC* 2.4 Hz
2	Kompas	Pin GPS	Pin 4	Ke APM 2.8, lalu ke *Receiver Radio Control* 2.4 Hz
3	ESC	Pin 3 Data Vcc Gnd	Pin 3 Data Vcc Gnd	Ke *Receiver Radio Control* 2.4 Hz
4	Motor Servo	Pin 1 Data Vcc Gnd	Pin 1 Data Vcc Gnd	Ke *Receiver Radio Control* 2.4 Hz
5	*Power Bank*	Pin USB	-	I = 1 Ampere
6	*Receiver Radio Control* 2.4 Hz	Pin 1,2,3,4, dan 8 Data	Data *Wireless*	Dari APM 2.8

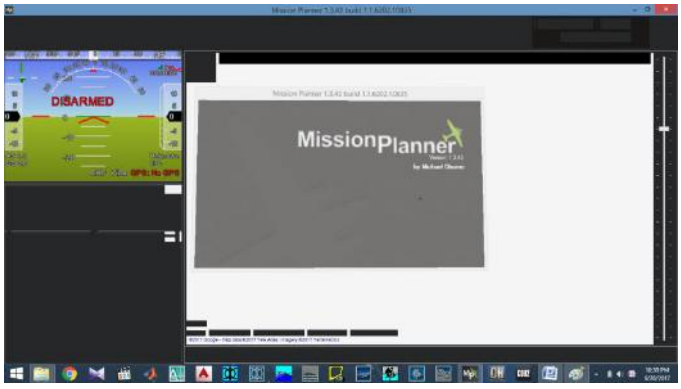
No	Komponen/Modul	*Input* APM 2.8	*Output*APM 2.8	Keterangan
		Vcc Gnd		Ke *Ground Control Station*
7	*Radio Telemetry* 915 MHz	Pin *Telemetry*	Data *Wireless*	Dari APM 2.8 Ke *Ground Control Station*



Gambar IV.32 Hasil Pemasangan Perangkat Autopilot

6. Penginstalan *SoftwareMission Planner*

Penginstalan *softwareMission Planner* versi 1.3.43 pada laptop Asus untuk digunakan pilot di *Ground Control Station.*



Gambar IV.33*SoftwareMission Planner*

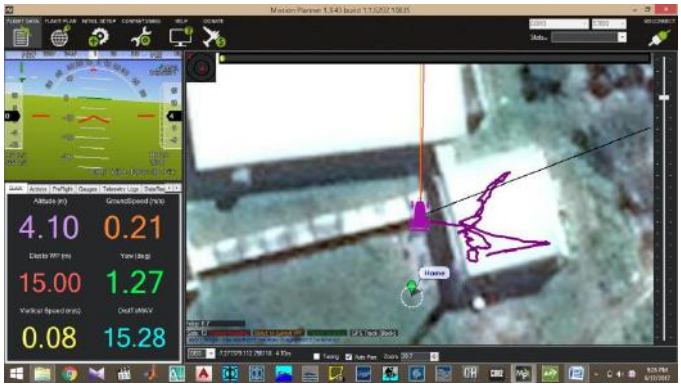
7. Pemasangan *Power Bank*

Peletakkan *power bank* diusahakan sedekat mungkin dengan APM 2.8untuk menghemat panjang kabel. Selain itu, *power bank* juga digunakan sebagai *ballast* pada purwarupasehingga diletakkan di ruangan bagian depan purwarupa.

8. Akses dan Kalibrasi

Mikrokontroler ArduPilot Mega 2.8, sebagai pusat pengendali USV yang diakses dan dikalibrasi terlebih dahulu oleh pilot di *Ground Control Station* (GCS) baik secara

langsung menggunakan kabel USB maupun secara tidak langsung menggunakan komunikasi *wireless* .APM 2.8akan diakses melalui *software Mission Planner* oleh pilot melalui laptop di *Ground Control Station* (GCS). Setelah itu, dikalibrasi untuk menentukan keakuratan koordinat GPS, keakuratan arah kompas, kecepatan *Electronic Speed Control* (ESC) dan Motor *Brushless,* dan pergerakan Motor Servo serta *rudder.*



Gambar IV.34 Akses ke *Software Mission Planner*

IV.5.3. Program pada ArduPilot Mega 2.8

Program pada ArduPilot Mega 2.8 menggunakan algoritma bahasa pemrograman C. Program ditulis di *Software Arduino* IDE, walaupun masih ada beberapa software lain yang sangat berguna selama pengembangan arduino. IDE atau *Integrated Development Environment* merupakan suatu program khusus untuk suatu komputer agar dapat membuat suatu rancangan atau sketsa program untuk modul Arduino. IDE arduino terdiri dari:

- Editor program sebuah *window* yang memungkinkan pengguna menulis dan mengedit program dalam bahasa processing
- *Compiler* Sebuah modul yang mengubah kode program menjadi kode biner bagaimanapun sebuah mikrokontroler tidak akan bisa memahami bahasa *processing*.
- *Uploader* Sebuah modul yang memuat kode biner dari komputer ke dalam memori didalam modul arduino.

Program yang digunakan untuk mengakses GPS, motor servo, ESC, dan kompas berdasarkan *library*dari APM 2.8 itu sendiri. Contoh algoritma program pada Arduino IDX dapat dilihat pada Gambar IV.35, IV.36, IV.37dan IV.38.Untuk lebih lengkapnya dapat dilihat pada halaman lampiran 2.

```
    if (!(mask & 1U<<(_channel-1))) {
        // not allowed
        return false;
    }
    channel = _channel;
    type = EVENT_TYPE_SERVO;

    start_time_ms  = 0;
    delay_ms    = _delay_ms / 2;
    repeat      = _repeat * 2;
    servo_value = _servo_value;
    update_events();
    return true;
}

bool AP_ServoRelayEvents::do_repeat_relay(uint8_t relay_num, int16_t _repeat, uint32_t _delay_ms)
{
    if (!relay.enabled(relay_num)) {
        return false;
    }
    type = EVENT_TYPE_RELAY;
    channel = relay_num;
    start_time_ms  = 0;
    delay_ms        = _delay_ms/2; // half cycle time
    repeat          = _repeat*2;  // number of full cycles
    update_events();
    return true;
}


/*
  update state for MAV_CMD_DO_REPEAT_SERVO and MAV_CMD_DO_REPEAT_RELAY
*/
void AP_ServoRelayEvents::update_events(void)
{
```

Gambar IV.35 Algoritma Program untuk Mengatur Servo

```
        extern const AP_HAL::HAL& hal;

        // parameters for the motor class
        const AP_Param::GroupInfo AP_MotorsMulticopter::var_info[] = {
            // 0 was used by TB_RATIO
            // 1,2,3 were used by throttle curve
            // 5 was SPIN_ARMED

            // @Param: YAW_HEADROOM
            // @DisplayName: Matrix Yaw Min
            // @Description: Yaw control is given at least this pwm in microseconds range
            // @Range: 0 500
            // @Units: PWM
            // @User: Advanced
            AP_GROUPINFO("YAW_HEADROOM", 6, AP_MotorsMulticopter, _yaw_headroom, AP_MOTORS_YAW_HEADROOM_DE

            // 7 was THR_LOW_CMP

            // @Param: THST_EXPO
            // @DisplayName: Thrust Curve Expo
            // @Description: Motor thrust curve exponent (from 0 for linear to 1.0 for second order curve)
            // @Range: 0.25 0.8
            // @User: Advanced
            AP_GROUPINFO("THST_EXPO", 8, AP_MotorsMulticopter, _thrust_curve_expo, AP_MOTORS_THST_EXPO_DEF

            // @Param: SPIN_MAX
            // @DisplayName: Motor Spin maximum
            // @Description: Point at which the thrust saturates expressed as a number from 0 to 1 in the
            // @Values: 0.9:Low, 0.95:Default, 1.0:High
            // @User: Advanced
            AP_GROUPINFO("SPIN_MAX", 9, AP_MotorsMulticopter, _spin_max, AP_MOTORS_SPIN_MAX_DEFAULT),

            // @Param: BAT_VOLT_MAX
            // @DisplayName: Battery voltage compensation maximum voltage
```

Gambar IV.36 Algoritma Program untuk Mengatur Motor *Brushless*

```
extern const AP_HAL::HAL& hal;

#if UBLOX_DEBUGGING
 # define Debug(fmt, args ...)  do {hal.console->printf("%s:%d: " fmt "\n", __FUNCTION__, __LINE__
#else
 # define Debug(fmt, args ...)
#endif

AP_GPS_UBLOX::AP_GPS_UBLOX(AP_GPS &_gps, AP_GPS::GPS_State &_state, AP_HAL::UARTDriver *_port) :
    AP_GPS_Backend(_gps, _state, _port),
    _step(0),
    _msg_id(0),
    _payload_length(0),
    _payload_counter(0),
    _class(0),
    _cfg_saved(false),
    _last_cfg_sent_time(0),
    _num_cfg_save_tries(0),
    _last_config_time(0),
    _delay_time(0),
    _next_message(STEP_PVT),
    _ublox_port(255),
    _have_version(false),
    _unconfigured_messages(CONFIG_ALL),
    _hardware_generation(UBLOX_UNKNOWN_HARDWARE_GENERATION),
    _new_position(0),
    _new_speed(0),
    _disable_counter(0),
    next_fix(AP_GPS::NO_FIX),
    _cfg_needs_save(false),
    noReceivedHdop(true),
    havePvtMsg(false)
{
    // stop any config strings that are pending
```

Gambar IV.37 Algoritma Program untuk Mengatur GPS

```
#include <AP_HAL/AP_HAL.h>
#include <AP_Notify/AP_Notify.h>
#include <GCS_MAVLink/GCS.h>

#include "AP_Compass.h"

extern AP_HAL::HAL& hal;

void
Compass::compass_cal_update()
{
    bool running = false;

    for (uint8_t i=0; i<COMPASS_MAX_INSTANCES; i++) {
        bool failure;
        _calibrator[i].update(failure);
        if (failure) {
            AP_Notify::events.compass_cal_failed = 1;
        }

        if (_calibrator[i].check_for_timeout()) {
            AP_Notify::events.compass_cal_failed = 1;
            cancel_calibration_all();
        }

        if (_calibrator[i].running()) {
            running = true;
        } else if (_cal_autosave && !_cal_saved[i] && _calibrator[i].get_status() == COMPASS_CAL_S
            _accept_calibration(i);
        }
    }

    AP_Notify::flags.compass_cal_running = running;

    if (is_calibrating()) {
```

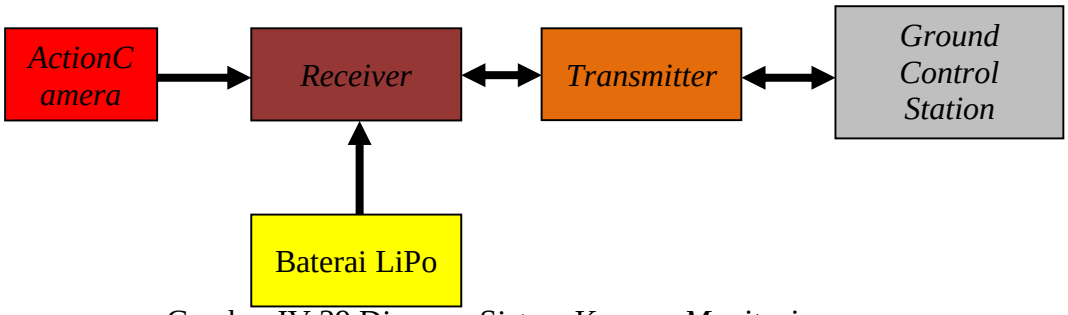
Gambar IV.38 Algoritma Program untuk Mengatur Kompas

IV.6. Sistem Kamera *Monitoring*

Sistem kamera *monitoring*adalah bagian dari sistem yang memungkinkan agar USV dapat melakukan tugas *monitoring* dengan menggunakan kamera yang dipasang pada USV dan dapat diakses dari sudut pandang pilot. Dengan kamera yang diletakkan pada USV tersebut, pilot dapat merasakan seolah-olah berada di dalam wahana tersebut dan melakukan *monitoring* maupun pengendalian wahana dengan mudah. Dengan adanya FPV maka operator dapat mengetahui arah, kondisi sekitar maupun lokasi yang akan dituju oleh USV.Sistem kamera *monitoring*USV ini didukung komponen berupakamera *action,* baterai,*Transmitter*,dan *Receiver*.Pengiriman *image data* dari kamera menuju *Ground Control Station* dilakukan secara *wireless.*

IV.6.1. Desain Sistem Kamera *Monitoring*

Berikut adalah diagram desain sistem autopilot untuk purwarupa tes model *Unmanned Surface Vehicle* yang ditunjukkan pada Gambar IV.35.



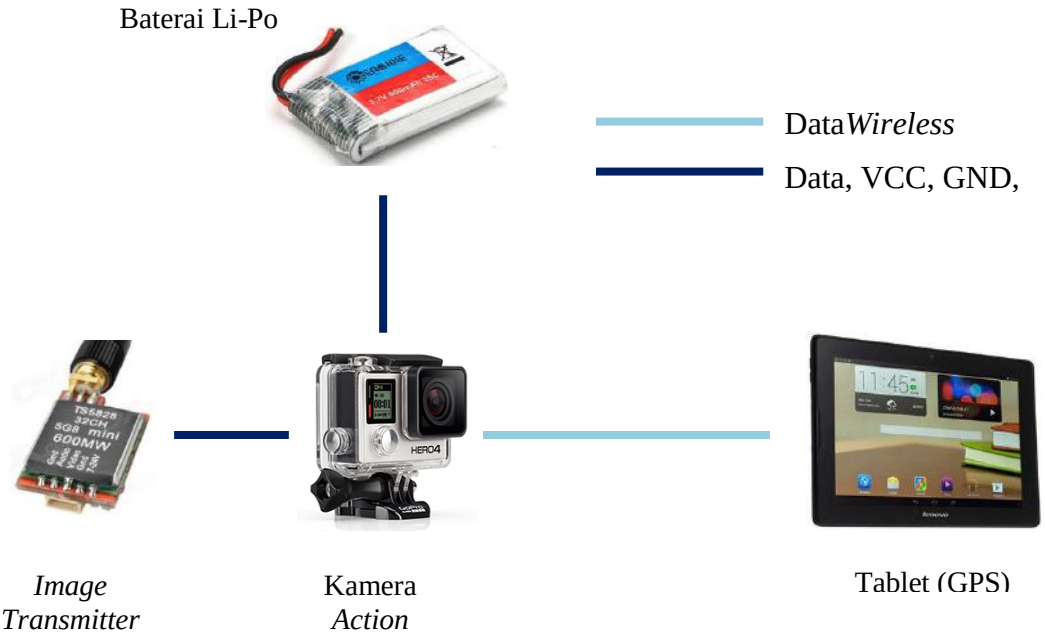
Gambar IV.39 Diagram Sistem Kamera *Monitoring*

Keterangan untuk diagram :

1. Kamera*Action*, sebagai komponen utama untuk melakukan *monitoring* pada purwarupa USV. Kamera *action* yang digunakan adalah GoPro Hero 4. Pada Kamera *action* dipasang pada *main mast* purwarupa USV.
2. *Transmitter*, berfungsi untuk mengirimkan citra gambar video dari USV ke *Ground Control Station* (GCS)secara *real time*, media komunikasi bersifat tanpa kabel *(wireless).* Menggunakan *wireless* 5.8G 600MW 48 Channel AV yang dipasang pada Kamera *action* GoPro Hero 4di *main mast* purwarupa USV.
3. *Ground Control Station* (GCS), sebagai pusat pengendalian USV oleh pilot. Perangkat yang digunakan adalah Tabletuntuk menampilkan citra gambar video dari kamera yang terdapat pada USV ke *Ground Control Station* (GCS) secara *real time.*

4. Baterai Li-Po Tiger 2S 7.4 Volt 660 mAh, sebagai catu daya untuk komponen *receiver* pada purwarupa USV . *Output* daya yang dihasikan adalah sebesar 7.4 volt.

Untuk *wiring diagram* antara kamera *action* dengan *transmitter* dan *Ground Control Station* adalah sebagai berikut :



Gambar IV.40*Wiring Diagram* Sistem Kamera *Monitoring*

IV.6.2. Pembuatan Sistem Kamera *Monitoring*

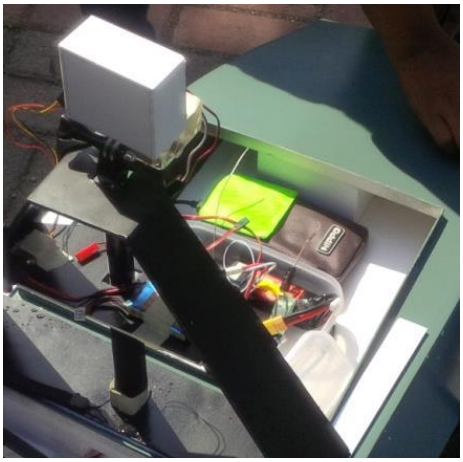
1. Pemasangan Kamera dan *Transmitter*

   Kamera dipasang di puncak main mast purwarupa USV. Kamera diletakkan tepat pada posisi tegak dan menghadap ke depan. Lalu dikunci dengan baut agar tidak bergeser ketika purwarupa USV mulai bergerak.

Gambar IV.41 Hasil Pemasangan Kamera

2. Pemasangan Baterai dan Kabel

Baterai dipasang di dekat dengan *transmitter* untuk menghemat panjang kabel. Pemasangan kabel dilakukan sesuai desain *wiring diagram.*



Gambar IV.42 Hasil Pemasangan Perangkat Sistem Kamera

3. Akses kamera pada *Ground Control Station* (Tablet)

Kamera GoPro Hero 4, sebagai alat *monitoring* USV harus terlebih dahulu oleh pilot di *Ground Control Station* (GCS) secara tidak langsung menggunakan komunikasi *wireless* .Hasil image data akan ditampilkan di *Ground Control Station* (GCS) melalui tablet.

Gambar IV.43 Akses Kamera pada GCS (Tablet)

BAB V
PENGUJIAN DAN ANALISA

V.1. Gambaran Umum

Pada Tugas Akhir kali ini, USV diimplementasikan untuk *monitoring* perairan. USV dikendalikan oleh seorang pilot menggunakan sebuah *remote control* ketika bergerak dalam mode manual dimana seorang pilot harus selalu *stand by* untuk mengontrol pergerakan kapal, di sisi lain pilot juga harus memantau hasil *monitoring*kapal melalu tablet. Selain itu USV juga dapat bergerak dalam metode autopilotsehingga seorang pilot tidak harus *standby* untuk mengontrol pergerakan kapal, namun kapal dapat begerak dengan sendirinya dan pilot hanya memantau hasil *monitoring*kapal serta dapat mengetahui pergerakan dari kapal melalui GPS yang ditampilkan pada *Mission Planner*. Oleh karena itu, dibuatlah sebuah sistem yang memungkinkan kapal untuk bergerak secara autopilot dengan menggunakan sistem navigasi GPS dan kompas yang menggunakan algoritma *waypoint.* Jadi USV dapat bergerak secara autopilot dengan dikendalikan secara autopilot menggunakan PC (*Personal Computer*) sebagai interface pengguna dimana komunikasinya menggunakan modul *wireless*, tanpa sepenuhnya menggunakan *remote control* dan bisa mengikuti jalur yang telah dibuat dengan bantuan perangkat GPS (*Global Positioning System*) dan dipandu oleh kompas yang telah terintegrasi dalam USV. Perangkat GPS akan menangkap sinyal dari satelit GPS yang menghasilkan koordinat *latitude* dan *longatitude* terhadap lokasi perangkat GPS tersebut berada. USV diprogram dengan koordinat *waypoint* yang dipilih dan akan melaju dengan kecepatan konstan pada saat menuju *waypoint*. *Waypoint* ini dapat berupa patok-patok perbatasan perairan yang dipasangi *buoy*.

Pada tahap awal, pilot akan membuat program *waypoint* untuk misi USV. Setelah itu pilot membawa USV dengan mode manual ke koordinat *home* sebagai *waypoint* pertama untuk melakukan misi *monitoring.* Ketika sudahpada koordinat *home*, maka pilot akan mengubah mode manual USV ke mode autopilot dan USV akan bergerak mengikuti *waypoint* yang sudah diprogram. Ketika USV ini bergerak menuju sasaran masing-masing, pilot yang berada di kapal induk akan memantau pergerakan tiap USV dari *Mission Planner*, dan memantau wilayah patroli berdasarkan hasil citra gambar video dari kamera *monitoring* yang dikirimkan oleh USV secara *real time*. Ketika terlihat ancaman atau hal yang mencurigakan pada citra gambar video

kamera *monitoring*, maka kapal induk dapat mengirimkan tim khusus untuk mengatasi ancaman tersebut via helikopter atau kapal cepat berawak lainnya. Pada saat USV yang sedang dalam mode autopilot tiba-tiba mulai terancam, operator dapat mengambil alih dan mengontrol USV dengan mengubah ke mode manual dengan bantuan *radio control.*

V.2. Pengujian Platform Purwarupa

Pengujian platform purwarupa adalah pengujian yang dilakukan untuk mengetahui kelayakan platform purwarupa USV sebagai tempat dimana sistem propulsi dan autopilot diinstal. Pengujian ini dilakukan beberapa tahap. Tahap pertama adalah setelah lambung platform purwarupa USV sudah jadi. Pengujian yang dilakukan yaitu uji kebocoran dan daya apung. Pengujian dilakukan di danau 8 ITS pada tanggal 25 Mei 2017.

Pengujian kebocoran dilakukan beberapa kali. Yang pertama adalah uji kebocoran tanpa menyentuh permukaan air danau. Caranya yaitu dengan membalikkan lambung kapal dan memposisikan lambung dalam keadaan miring sehingga kapal pada posisi vertikal dengan sudut kemiringan sekitar 45^{o}. Lalu disiramkan dengan air secara perlahan pada seluruh bagian bawah lambung secara terus menerus. Lalu dicek apakah pada bagian dalam lambung ada air yang menetes atau tidak. Jika ada, maka harus segera ditambal menggunakan fiber. Jika tidak ada, maka dilanjutkan ke uji kebocoran kedua.

Uji kebocoran yang kedua yaitu dengan menyentuh permukaan air danau. Caranya yaitu dengan meletakkan lambung kapal di permukaan air danau dilanjutkan menekan lambung kapal hingga hampir tenggelam seluruhnya. Lalu dicek apakah pada bagian dalam lambung ada air yang masuk atau tidak. Jika ada, maka harus segera ditambal menggunakan fiber. Jika tidak ada, maka dilanjutkan ke uji daya apung.

Setelah uji kebocoran, dilanjutkan dengan uji daya apung. Uji daya apung dilakukan dengan memberikan beban berupa balok paving pada lambung USV. Setelah itu kapal dimaju-mundurkan untuk memastikan bahwa kapal dapat mengapung dengan baik. Jika lambung kapal sudah diuji daya apung maka pengerjaan platform purwarupa dapat dilanjutkan untuk membuat dek, *main mast,* dan lainnya.

Gambar V.1 Uji Kebocoran dan Daya Apung

Pengujian platform purwarupa dilanjutkan ke tahap selanjutnya yaitu uji kestabilan. Pengujian dilakukan di danau 8 ITS pada tanggal 11 Juni 2017. Purwarupa harus terlebih dahulu diberi beban hingga mencapai garis sarat. Uji kestabilan dilakukan dengan menekan ujung depan purwarupa, lalu dilepas. Jika posisi purwarupa kembali seperti awal, maka dapat dinyatakan stabil. Lalu uji ini diulang dengan menekan masing-masing bagian ujung belakang *starboard,* dan *portside.* Jika sudah kembali pada posisi semula, maka purwarupa sudah dapat dinyatakan stabil.

Pengujian dilanjutkan dengan menjalankan kapal secara manual dengan memberi beban batu pada bagian ujung depan. Jika purwarupa dapat berjalan dengan stabil, maka dapat disimpulkan lolos pengujian ini. Uji ini diulang dengan memberi beban batu masing-masing bagian ujung belakang, lalu *starboard,* dan terakhir *portside.* Jika purwarupa dapat berjalan dengan stabil, maka dapat disimpulkan lolos pengujian ini.

(a)

(b)

Gambar V.2 (a) dan (b) Uji Kestabilan

Berikut adalah hasil pengujian platform purwarupa yang ditunjukkan pada Tabel V.1. Pengujian yang dilakukan yaitu uji kebocoran, daya apung, dan kestabilan. Hasil uji kebocoran adalah baik, dalam arti bahwa lambung purwarupa tidak mengalami kebocoran. Hasil uji daya apung juga menunjukkan bahwa purwarupa dapat mengapung dengan baik. Hasil uji kestabilan juga menunjukkan purwarupa dalam kondisi stabil ketika muatannya sesuai dengan garis sarat air. Sehingga dapat disimpulkan bahwa platform purwarupa lolos semua uji purwarupa dan dapat dilanjukan ke pengerjaan berikutnya.

Tabel V.1 Hasil Pengujian Platform Purwarupa

No.	Parameter Pengujian	Hasil	Keterangan
1	Kebocoran	Baik	Tidak ada kebocoran
2	Daya Apung	Baik	Purwarupa dapat mengapung dengan baik
3	Kestabilan	Baik	Stabil pada saat sesuai garis sarat

V.3. Pengujian Sistem Propulsi

Pengujian sistem propulsi adalah pengujian yang dilakukan untuk mengetahui fungsional sistem propulsi yang terdapat pada purwarupa USV. Pengujian sistem propulsi purwarupa USV dilakukan setelah lambung platform purwarupa USV sudah selesai diuji dan komponen sistem propulsi sudah dipasang. Pengujian yang dilakukan yaitu uji fungsi pengendalian mode manual,fungsi *motor, rudder, propeller,*dan pengujian kecepatan pada purwarupa USV. Pengujian dilakukan di danau 8 ITS pada tanggal 11 Juni 2017.

Pengujian fungsi pengendalian mode manual yaitu pengujian untuk mengetahui bahwa *Radio Contro*l 2.4 Hz di *Ground Control Station* dapat mengakses purwarupa USV atau tidak. Jika sudah dapat diakses, maka dapat dilakukan uji fungsi motor*brushless, rudder,* dan *propeller* yang dikendalikan lewat *Radio Control* 2.4 Hz.

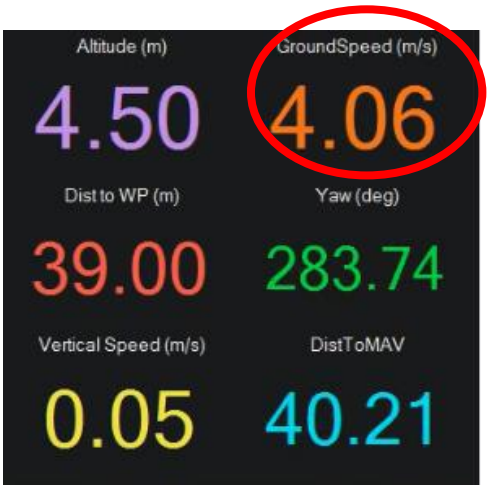
Pengujian fungsi motor*brushless, rudder,* dan *propeller* yaitu pengujian untuk mengetahui fungsionalitas dari komponen-komponen yang digunakan. Pengujian ini dilakukan dengan cara mencoba tiap-tiap komponen secara berulang kali. Sebelum diuji di dalam danau, komponen akan terlebih dahulu diuji di darat. Jika sudah bekerja dengan baik, maka pengujian dianjutkan dengan pengujian di dalam danau.

Untuk pengujian di dalam danau, purwarupa harus diposisikan statis dengan cara ditahan. Lalu komponen motor*brushless, rudder,* dan *propeller* diuji satu per satu sesuai fungsinya. Jika sudah bekerja dengan baik, maka pengujian dianjutkan dengan pengujian kecepatan pada purwarupa USV di danau. Pilot akan mengendalikan purwarupa USV secara

manual dan dijalankan hingga mencapai kecepatan maksimalnya. Untuk mengetahui kecepatan maksimalnya, maka pada purwarupa dipasang APM yang terhubung ke GCS untuk menerima informasi kecepatannya. Hail kecepatan tertingginya adalah 4.06 m/s atau 7.892 knot.

(a) (b)

Gambar V.3 (a) dan (b) Uji Propulsi



Gambar V.4 Kecepatan Tertinggi Purwarupa USV

Berikut adalah hasil pengujian platform purwarupa yang ditunjukkan pada Tabel V.2. Pengujian yang dilakukan yaitu uji fungsi pengendalian mode manual, fungsi motor*brushless, rudder, propeller,*dan pengujian kecepatan pada purwarupa USV. Hasil uji fungsi pengendalian mode manual adalah baik, dalam arti bahwa purwarupa USV dapat dikendalikan oleh pilot melalui *Radio Control* 2.4 Hz. Hasil uji fungsi motor*brushless, rudder, propeller*juga menunjukkan bahwa fungsi masing-masing komponen dapat bekerja dengan baik. Hasil uji kecepatan juga menunjukkan bahwa purwarupa dapat berjalan dengan kecepatan yang optimal. Sehingga dapat disimpulkan bahwa platform purwarupa lolos semua uji purwarupa dan dapat dilanjukan ke pengerjaan berikutnya.

Tabel V.2 Hasil Pengujian Sistem Propulsi

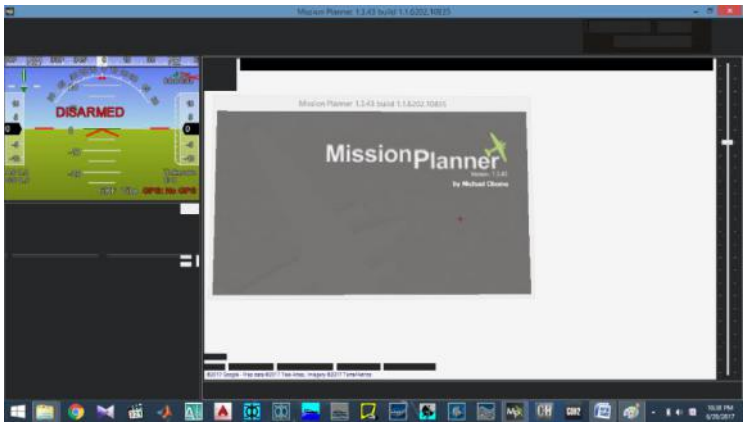
No.	Parameter Pengujian	Hasil	Keterangan
1	Fungsi Pengendalian Mode Manual	Baik	Pengendalian Mode Manual dengan baik
2	Fungsi Motor *Brushless*	Baik	Motor *Brushless*dapat bekerja dengan baik
3	Fungsi *Rudder*	Baik	*Rudder* dapat bekerja dengan baik
4	Fungsi *Propeller*	Baik	*Propeller* dapat bekerja dengan baik
5	Pengujian Kecepatan	Baik	Kecepatan maksimal mencapai 4.06 m/s atau 7.892 knot

V.4. Pengujian Sistem Autopilot

Pengujian sistem autopilot adalah pengujian yang dilakukan untuk mengetahui fungsional sistem autopilot yang terdapat pada purwarupa USV. Pengujian sistem autopilot purwarupa USV dilakukan setelah lambung platform dan komponen sistem propulsi purwarupa USV sudah selesai diuji. Pengujian yang dilakukan yaitu pengujian aplikasi *Mission Planner*dan ArduPilot Mega 2.8, uji fungsi sensor GPS dan kompas, uji fungsi sistem autopilot mode manual, dan uji fungsi sistem autopilot mode autopilot. Pengujian dilakukan di danau 8 ITS pada tanggal 20 Juni 2017-2 Juli 2017.

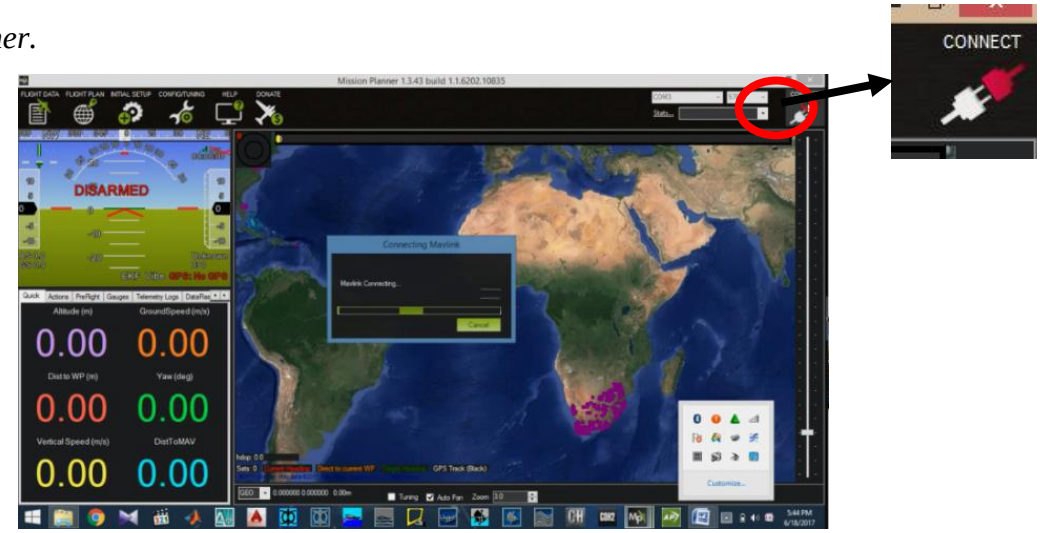
V.4.1. Pengujian Konektivitas Aplikasi *Mission Planner*dan ArduPilot Mega 2.8

Pengujian ini dilakukan untuk mengetahui konektivitas dan fungsionalitas dari aplikasi yang telah dibuat dan akses ke mikrokontroler ArduPilot Mega 2.8. Pengujian ini dimulai dengan cara memulai aplikasi, kemudian mengakses ArduPilot Mega 2.8 dengan *software Mission Planner*, dan memulai komunikasi dengan USV.



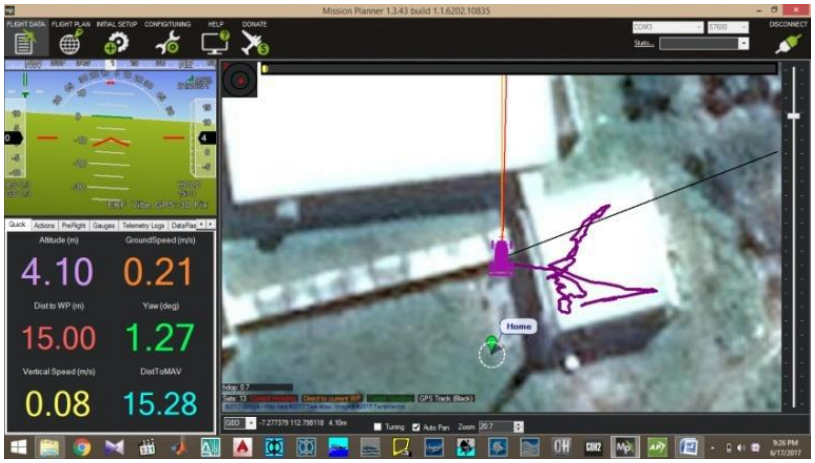
Gambar V.5 Memulai Aplikasi *Mission Planner*

Untuk pengujian aplikasi dan akses ArduPilot Mega 2.8 dengan *software Mission Planner*dilakukan di Gedung Robotika ITS pada tanggal 20 Juni 2017. Tidak ada kendala dalam membuka aplikasi *software Mission Planner.* Kemudian dilanjutkan dengan mengakses ArduPilot Mega 2.8 dengan *software Mission Planner*. Ada beberapa parameter yang harus dimasukkan di *software Mission Planner* supaya dapat terhubung ke APM 2.8 seperti saluran komunikasi, kode antar perangkat, dan lainnya. Setelah parameter diatas dapat ditemukan, maka kedua perangkat dapat dihubungkan dengan mengklik tombol '*connect'* pada *software Mission Planner.*



Gambar V.6 Mengakses APM 2.8 dengan Menggunakan *Mission Planner*

Jika *software Mission Planner*sudah dapat mengakses ArduPilot Mega 2.8, maka akan muncul tampilan lokasi USV seperti tampak pada gambar V.7 dibawah ini. Dari tampilan ini, berarti antar kedua perangkat sudah dapat berkomunikasi walaupun belum dapat mengakses keseluruhan sistem pada purwarupa. Namun hanya berupa saluran komunikasi dankode antar kedua perangkat saja.



Gambar V.7 Memulai Komunikasi dengan USV

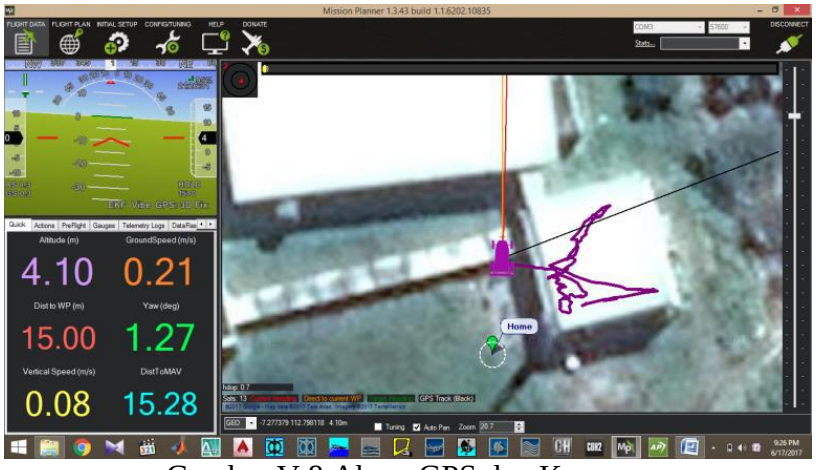
Berikut adalah hasil pengujian konektivitas aplikasi *Mission Planner*dan ArduPilot Mega 2.8yang ditunjukkan pada Tabel V.3.

Tabel V.3 Hasil Pengujian Konektivitas Aplikasi *Mission Planner* dengan APM 2.8

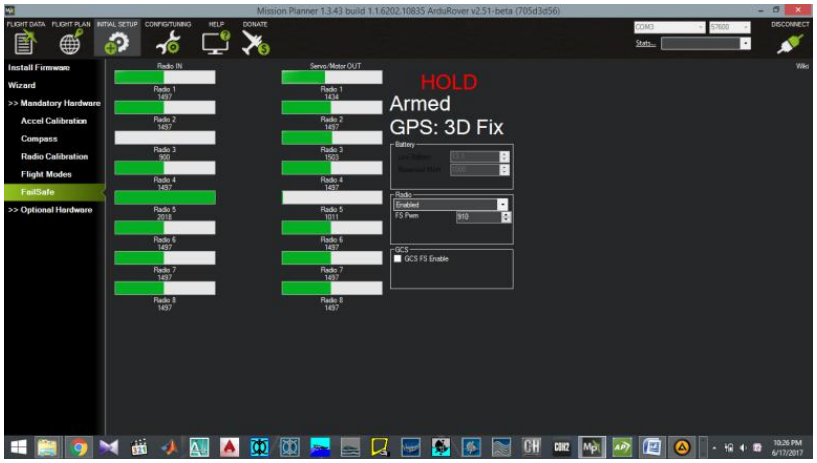
Pengujian	Banyak Pengujian	Berhasil	Gagal	Hasil
Mengakses APM 2.8 dengan *software Mission Planner*	20	15	5	Berhasil
Komunikasi antar kedua perangkat	20	18	2	Berhasil

V.4.2. Pengujian Fungsi Sensor GPS dan Kompas

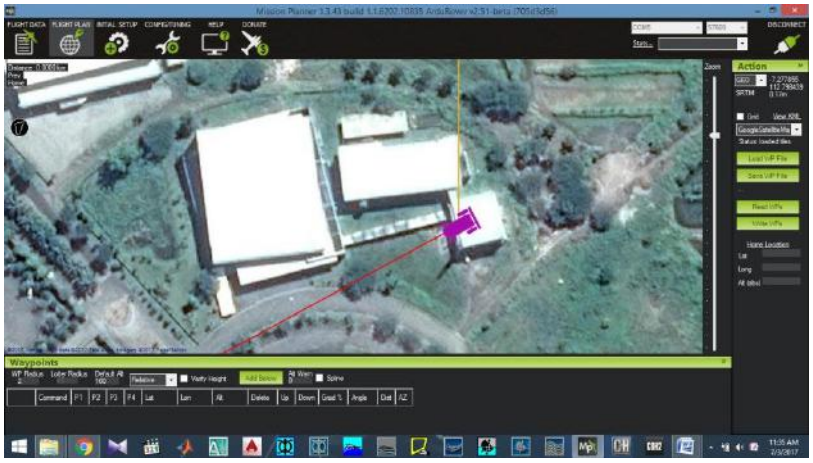
Pengujian ini dilakukan untuk mengetahui fungsionalitas dari sensor GPS dan Kompas pada purwarupa yang terhubung ke mikrokontroler ArduPilot Mega 2.8 dan diakses melalui *software Mission Planner*. Pengujian ini dimulai dengan kalibrasi GPS dan kompas sehingga didapatkan hasil arah GPS dan kompas, lalu dilanjutkan dengan uji coba daratsensor GPS dan kompas. Kalibrasi GPS dan kompas dilakukan dengan memasukkan beberapa parameter di *software Mission Planner* seperti *longitude*, *latitude*, *altitude* arah utara, selatan, barat, timur, ID, saluran pin pada APM 2.8, dan lainnya. Pada tampilan gambar V.8, GPS ditandai dengan garis berwarna hitam, sementara kompas ditandai dengan garis berwarna merah. Untuk warna ungu adalah posisi USV. Terlihat pada gambar, kalibrasi masih belum dilakukan sehingga belum didapatkan koordinat GPS yang pasti.



Gambar V.8 Akses GPS dan Kompas



Gambar V.9 Kalibrasi GPS dan Kompas


Gambar V.10 Hasil Kalibrasi Sensor GPS dan Kompas

Setelah parameter-parameter diatas dapat ditemukan, maka kedua sensor diuji dengan melakukan uji coba darat. Uji coba darat dilakukan dengan menginput *waypoint* pada *software Mission Planner*¸ kemudian purwarupa USV dibawa dengan cara diangkat dan berjalan menuju masing-masing koordinat. Pengujian ini dilakukan berulang-ulang untuk memastikan sensor GPS dan kompas dapat bekerja dengan baik. Jika *Mission Planner* dapat membaca koordinat GPS dan arah kompas dengan tepat, maka uji coba darat dinyatakan berhasil dan purwarupa USV siap untuk uji coba di air. Namun pada uji coba darat ini, tingkat error GPS belum bisa dicari kerena purwarupa USV masih dikendalikan secara manual. Hasil uji coba darat dapat dilihat pada gambar V.10. Terlihat ada 2 warna garis yang berbeda yaitu kuning dan ungu. Garis kuning lurus menjelaskan lintasan yang seharusnya ditempuh oleh purwarupa USV. Sementara pada garus ungu yang tidak lurus menjelaskan lintasan yang ditempuh oleh purwarupa USV itu sendiri. Ketidaklurusan lintasan diakibatkan oleh ketidakstabilan medan yang ditempuh oleh tim penguji.

Gambar V.11 Uji Coba Darat Sensor GPS dan Kompas

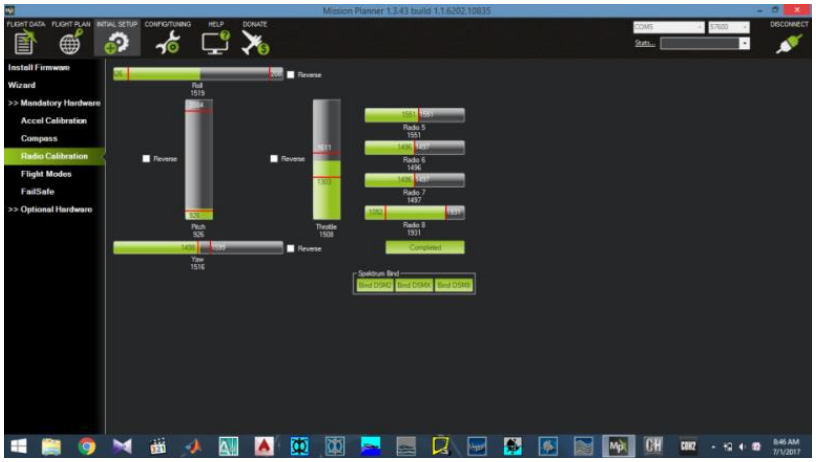
Berikut adalah hasil pengujian sensor GPS & kompasyang ditunjukkan pada Tabel V.4.

Tabel V.4 Hasil Pengujian Sensor GPS dan Kompas

Pengujian	Banyak Pengujian	Berhasil	Gagal	Hasil
GPS	20	7	13	Berhasil
Kompas	20	9	11	Berhasil

V.4.3. Pengujian Fungsi Sistem AutopilotMode Manual

Pengujian ini dilakukan untuk mengetahui fungsionalitas dari sensor GPS dan kompas padamode manual purwarupa USV. Pengujian ini dimulai dengan kalibrasi *Electronic Speed Control* (ESC), motor b*rushless* dan motor servo untuk *rudder* sehingga didapatkan kecepatan motor dan sudut belok *rudder* yang optimal.Kalibrasi tiap komponen tersebut dilakukan dengan memasukkan parameter-parameter di *software Mission Planner* seperti kecepatan terendah, kecepatan tertinggi, sudut belok minimal, sudut belok maksimal, mode tiap *channel* pada *Radio Control* 2.4 Hz, dan lainnya. Kalibrasi ESC seperti tampak pada tampilan gambar V.12.

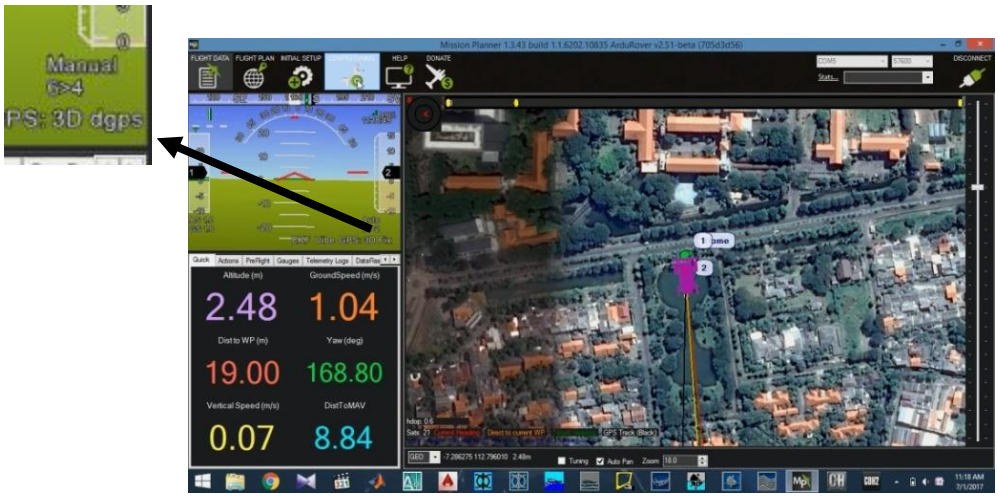


Gambar V.12 Kalibrasi ESC

Untuk membedakan mode manual dan autopilot, maka pin flight mode dikalibrasi supaya tiap switch pada Radio Control 2.4 Hz memiliki frekuensi yang berbeda. Hasil kalibrasi dapat dilihat pada gambar V.13 di bawah ini.

Gambar V.13Kalibrasi Mode Tiap Pin pada*Radio Control* 2.4 Hz dan*Mission Planner*

Setelah parameter-parameter diatas telah dikalibrasi dan dapat diakses, maka semua komponen sistem autopilot diuji coba di air. Untuk komponen yang diuji adalah sensor GPS, kompas, motor *brushless, rudder,* dan ESC. Langkah pertama adalah menentukan tujuan purwarupa USV akan bergerak. Setelah itu memindahkan *switch* pada *Radio Control* 2.4 Hz pada mode manual. Kemudian purwarupa USV diarahkan menuju koordinat yang telah ditentukan. Pada pengujian ini akan didapatkan pergerakan USV, koordinat USV, arah kompas yang dituju, kecepatan yang dihasilkan, dan manuver yang dilakukan oleh purwarupa USV. Pengujian ini dilakukan berulang-ulang untuk memastikan sensor GPS dan kompas, motor *brushless, rudder,* dan ESC dapat bekerja dengan baik. Pengujian ini juga dilakukan untuk menampilkan lokasi USV pada layar monitor GCS secara *real time.*Hasil Pengujian dapat dilihat pada table

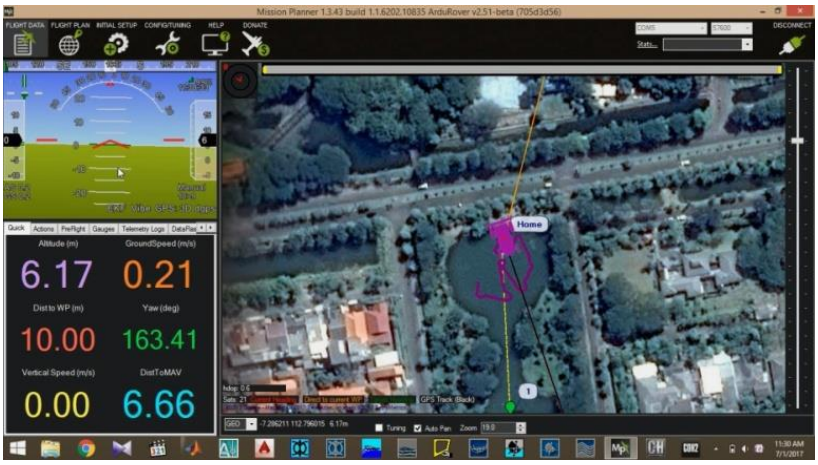


Gambar V.14Pengujian Sistem Autopilot Mode Manual

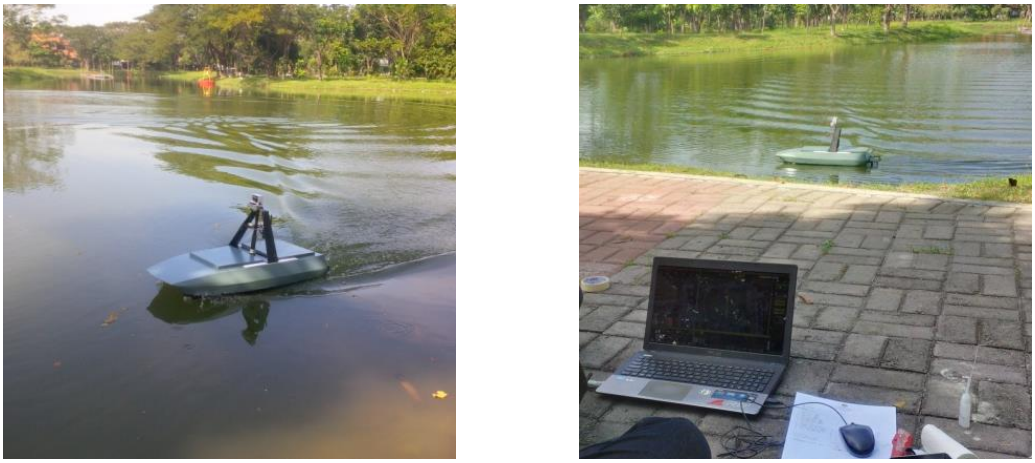
Jika *Mission Planner* dapat membaca koordinat GPS, arah kompas, dan kecepatan maksimal dengan tepat sesuai hasil kalibrasi, maka uji coba purwarupa USV dinyatakan berhasil. Namun pada uji coba ini, tingkat error GPS belum bisa dicari kerena purwarupa USV masih dikendalikan dalam mode manual. Hasil uji coba darat dapat dilihat pada gambar V.15. Terlihat ada 5 warna garis yang berbeda yaitu merah, hitam, oranye, kuning dan ungu. GPS ditandai dengan garis berwarna hitam, sementara kompas ditandai dengan garis berwarna merah. Untuk warna ungu adalah posisi USV. Garis kuning dan garis oranye menjelaskan lintasan yang akan ditempuh oleh purwarupa USV tetapi hanya berfungsi ketika mode *autopilot*saja. Sementara pada garis ungu yang tidak lurus menjelaskan lintasan yang ditempuh oleh purwarupa USV itu sendiri yang diarahkan secara manual oleh pilot.

Tabel V.5Hasil Parameter Pengujian Sistem Autopilot Mode Manual

No.	Parameter Pengujian	Hasil	Keterangan
1	Pergerakan USV	Baik	Dapat dipantau melalui GCS
2	Koordinat USV	Baik	Dapat dipantau melalui GCS
3	Arah Kompas USV	Baik	Dapat dipantau melalui GCS
4	Kecepatan USV	Baik	Dapat dipantau melalui GCS
5	Manuver USV	Baik	Dapat dipantau melalui GCS



Gambar V.15 Lintasan yang Dihasilkan dari Pergerakan USV (Garis Ungu)



(a) (b)

Gambar V.16 Uji Coba di Air pada Mode Manual dan (b) Tampilan USV dari GCS

V.4.4. Pengujian Fungsi Sistem AutopilotMode Autopilot

Pengujian ini dilakukan untuk mengetahui fungsionalitas dari sensor GPS dan kompas, ESC, motor *brushless,*dan motor servo untuk *rudder* padamode *autopilot* purwarupa USV. Pengujian ini dimulai dengan kalibrasi *Electronic Speed Control* (ESC), motor b*rushless* dan motor servo untuk *rudder* sehingga didapatkan kecepatan motor dan sudut belok *rudder* yang optimal. Kalibrasi tiap komponen tersebut dilakukan dengan memasukkan parameter-parameter di *software Mission Planner* seperti kecepatan terendah, kecepatan tertinggi, sudut belok minimal, sudut belok maksimal, mode tiap *channel* pada *Radio Control* 2.4 Hz, dan lainnya.Setelah parameter-parameter diatas telah dikalibrasi dan dapat diakses, maka semua komponen sistem autopilot diuji coba di air.

Untuk komponen yang diuji adalah sensor GPS, kompas, motor *brushless, rudder,* dan ESC. Langkah pertama adalah menentukan tujuan purwarupa USV akan bergerak dengan menginput titik-titik *waypoint*. Mode yang digunakan masih mode manual. Kemudian purwarupa USV digerakkan secara manual menuju koordinat *home* yang telah ditentukan. Ketika sudah di titik home, *switch* pada *Radio Control* 2.4 Hzdipindahkan pada mode auto. Padapengujian ini akan didapatkan pergerakan USV, koordinat yang dituju, arah kompas yang dituju, kecepatan yang dihasilkan, dan belokan yang dilakukan oleh purwarupa USV. Pengujian ini dilakukan berulang-ulang dan menggunakan bermacam-macam jenis lintasan untuk memastikan sensor GPS dan kompas, motor *brushless, rudder,* dan ESC dapat bekerja dengan baik. Pengujian ini juga dilakukan untuk menampilkan lokasi USV pada layar monitor GCS secara *real time.*

Gambar V.17 Pengujian Sistem Autopilot Mode *Autopilot*

Jika *Mission Planner* dapat membaca koordinat GPS, arah kompas, dan kecepatan maksimal dengan tepat sesuai hasil kalibrasi, maka uji coba purwarupa USV dinyatakan berhasil. Pada uji coba ini, tingkat error GPS sudah bisa dicari kerena purwarupa USV sudah dikendalikan dalam mode *autopilot*. Hasil uji coba darat dapat dilihat pada gambar V.18. Terlihat ada 5 warna garis yang berbeda yaitu merah, hitam, oranye, kuning dan ungu. GPS ditandai dengan garis berwarna hitam, sementara kompas ditandai dengan garis berwarna merah. Untuk warna ungu adalah posisi USV. Garis kuning adalah garis lintasan yang telah dibuat, garis oranye menjelaskan lintasan yang akan dituju oleh purwarupa USV. Sementara pada garis ungu yang tidak lurus menjelaskan lintasan yang ditempuh oleh purwarupa USV itu sendiri yang diarahkan secara manual oleh pilot.

Gambar V.18 Hasil Lintasan Pengujian Sistem Autopilot Mode *Autopilot*

Pengujian lintasan dilakukan pada 4 tipe lintasan dengan masing-masing memiliki titik-titik *waypoint* yang berbeda. Lintasan pertama adalah lintasan dengan 2 titik *waypoint*, dengan jarak 15 meter. Lintasan kedua adalah lintasan dengan 2 titik *waypoint*, dengan jarak 100 meter.Lintasan ketiga adalah lintasan dengan 4 titik.Dan lintasan keempat adalah lintasan dengan 6 titik.

Pada pengujian lintasan 1, lintasan yang ditempuh ditunjukkan pada gambar V.19. Lintasan ini memiliki 2 titik *waypoint*yang memiliki jarak ±15 meter. Setelah dilakukan pengujian, purwarupa USV bergerak menuju titik *waypoint*yang telah dibuat. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya

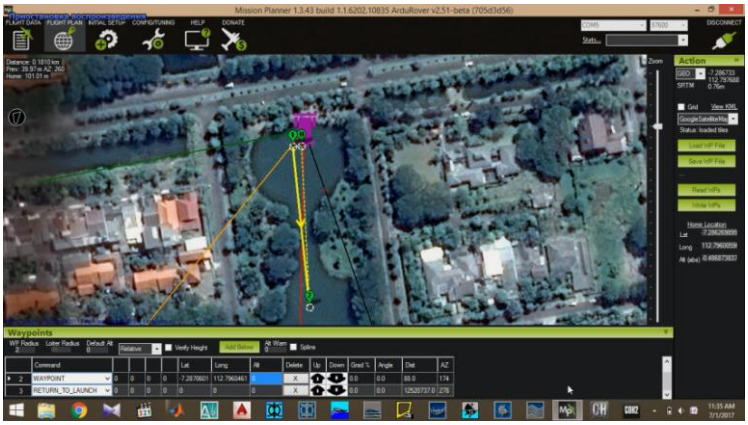
Gambar V.19 Lintasan 1

Pada pengujian lintasan 1, koordinat *waypoint* adalah ditampilkan pada table V.6.

Tabel V.6 Koordinat Waypoint 1

Jarak Titik *Waypoint* (m)	**Koordinat**
Home	Latt -7.286202055Long 112.7960421
1	Latt -7.2861981Long 112.7960046
2	Latt -7.2864455Long 112.7960113

Pada pengujian lintasan 2, lintasan yang ditempuh ditunjukkan pada gambar V.20. Lintasan ini memiliki 2 titik *waypoint*yang memiliki jarak ±100 meter. Setelah dilakukan pengujian, purwarupa USV bergerak menuju titik *waypoint*yang telah dibuat. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya.



Gambar V.20 Lintasan 2

Pada pengujian lintasan 1, koordinat *waypoint* adalah ditampilkan pada table V.7.

Tabel V.7 Koordinat Waypoint 2

Jarak Titik *Waypoint* (m)	**Koordinat**
Home	Latt -7.286269899Long 112.7960059
1	Latt -7.2862366Long 112.7960756
2	Latt -7.2870601Long 112.7960461

Pada pengujian lintasan 3, lintasan yang ditempuh ditunjukkan pada gambar V.21. Lintasan ini memiliki 4 titik *waypoint*yang memiliki jarak dari titik 1 ke 2 dan 3 ke 4 adalah sebesar 15 meter, sementra antara titik 2 ke 3 dan titik 4 ke 1 adalah 10 meter. Setelah dilakukan pengujian, purwarupa USV bergerak menuju titik *waypoint*yang telah dibuat. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya.

Gambar V.21 Lintasan 3

Pada pengujian lintasan 3, koordinat *waypoint* adalah ditampilkan pada table V.8.

Tabel V.8 Koordinat Waypoint 3

Jarak Titik *Waypoint* (m)	**Koordinat**	
Home	Latt -7.2862366	Long 112.7960756
1	Latt -7.2862260	Long 112.7960126
2	Latt -7.2863364	Long 112.7958551
3	Latt -7.2864349	Long 112.7959697
4	Latt -7.2863098	Long 112.7961172

Pada pengujian lintasan 4, lintasan yang ditempuh ditunjukkan pada gambar V.22. Lintasan ini memiliki 6 titik *waypoint*. Setelah dilakukan pengujian, purwarupa USV bergerak menuju titik *waypoint*yang telah dibuat. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya.

Gambar V.22 Lintasan 4

Pada pengujian lintasan 4, koordinat *waypoint* adalah ditampilkan pada table V.9.

Tabel V.9 Koordinat *Waypoint* 4

Jarak Titik *Waypoint* (m)	**Koordinat**	
Home	Latt -7.2862639	Long 112.7960045
1	Latt -7.2862692	Long 112.7959958
2	Latt -7.2863271	Long 112.7958832
3	Latt -7.2864548	Long 112.7959053
4	Latt -7.2864748	Long 112.7960448
5	Latt -7.2863903	Long 112.7951323
6	Latt -7.2862312	Long 112.7961119

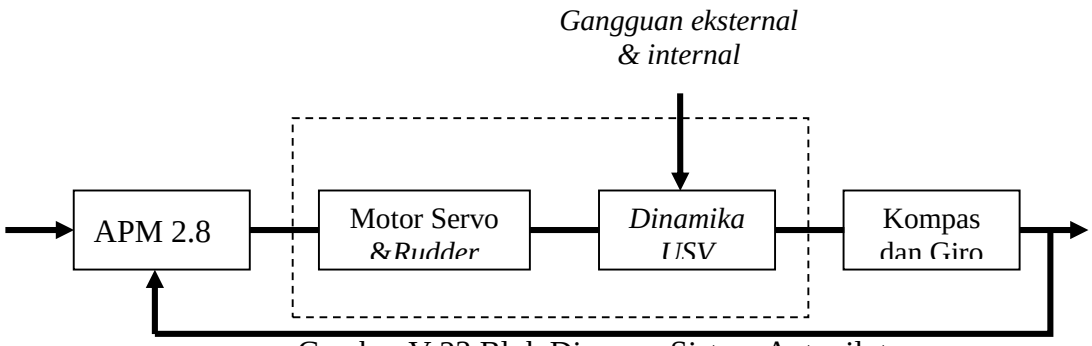
Dari 4 pengujian untuk sistem autopilot mode *autopilot,* semuanya sudah tepat menuju koordinat masing-masing sesuai dengan *waypoint* yang diinput oleh pilot melalui GPS. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya.

Hasil dari pengujian 4 lintasan, dapat disimpulkan bahwa USV bergerak menuju koordinat *waypoint* dengan tepat sesuai dengan arahan dari sistem navigasi GPS. Namun pergerakan purwarupa USV tidaklah lurus seperti lintasan yang seharusnya (ditunjukkan oleh garis kuning), namun berbelok-belok. Hal ini disebabkan oleh pengaruh gaya luar dari purwarupa USV yaitu akibat angin dan gelombang pada air danau. Penyebab lainnya adalah gaya dari dalam seperti ketidakseimbangan putaran kecepatan motor induk akan memberikan pengaruh dalam pergerakannya. Untuk hasil pengujian dapat dilihat pada table sebagai berikut:

Tabel V.10 Hasil pengujian 1 – 4

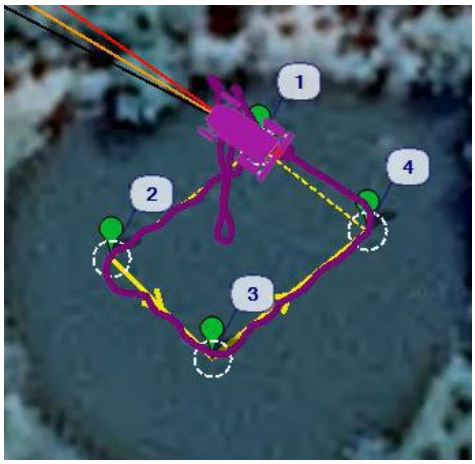
No.	Parameter Pengujian	Hasil	Keterangan
1	Lintasan 1 (2 *waypoint* jarak pendek ±15 m)	Baik	Tepat menuju *waypoint*, tetapi lintasan yang ditempuh tidak sesuai lintasan yang diinginkan
2	Lintasan 2 (2 *waypoint* jarak pendek ±100 m)	Baik	Tepat menuju *waypoint*, tetapi lintasan yang ditempuh tidak sesuai lintasan yang diinginkan
3	Lintasan 3 (4 *waypoint*)	Baik	Tepat menuju *waypoint*, tetapi lintasan yang ditempuh tidak sesuai lintasan yang diinginkan
4	Lintasan 4 (6 *waypoint*)	Baik	Tepat menuju *waypoint*, tetapi lintasan yang ditempuh tidak sesuai lintasan yang diinginkan

Dalam pergerakan USV, sistem navigasi pada GPS tidak bekerja sendirian. Tetapi bekerja bersamaan dengan sistem kontrol yang memeriksa fungsi dari autopilot USV. Komponen utama yang bekerja pada sistem kontrol adalahsensor kompas. Sistem navigasi GPS dan sistem kontrol dari kompas diolah oleh mikrokontroler APM 2.8 untuk menggerakkan motor servo dan *rudder* untuk bergerak menuju arah titik *waypoint*.Jadi, sistem autopilot akan bekerja berdasarkan input data dari sistem navigasi GPS dan diolah menggunakan bahasa pemrograman C dari *Arduino* IDE untuk diinput lagi ke sistem kontrol. Sistem kontrol menggunakan data dari koordinat GPS dan sudut *error*kompas untuk dapat menyelesaikan algoritma untuk akses ke motor servo dan *rudder.*



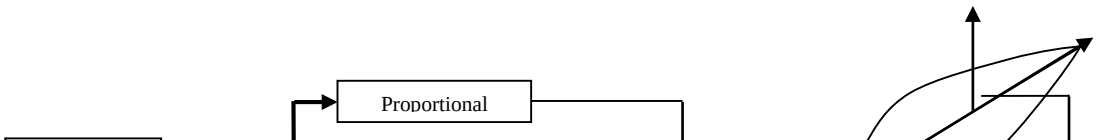
Gambar V.23 Blok Diagram Sistem Autopilot

Berdasarkan blok diagram diatas, mikrokontroler APM 2.8 akan mengolah data dari koordinat GPS yang digunakan untuk menggerakkan motor servo dan *rudder* sehingga akan menggerakkan USV menuju titik tuju. Dalam perjalanannya, pergerakan USV akan mendapat gangguan yang disebabkan oleh faktor-faktor eksternal seperti gelombang dan angin, dan faktor internal berupa hasil kerja ESC dan motor *brushless.* Faktor-faktor pengganggu ini akan menyebabkan USV bergerak menjauhi lintasan awal yang seharusnya langsung menuju titik tuju. Tetapi, walaupun pergerakan USV terganggu dengan menjauhi lintasannya, USV akan kembali lagi ke dalam lintasannya dan tetap bergerak menuju titik tuju. Gangguan ini akan terjadi terus menerus tetapi USV akan bergerak lagi menuju lintasannya sehingga lintasan yang dilalui USV tidaklah garis lurus, tetapi berkelok-kelok seperti yang tampak pada gambar V.24 dibawah ini. Garis kuning menggambarkan lintasan yang seharusnya ditempuh oleh USV, garis ungu menggambarkan lintasan yang ditempuh oleh USV. Bagaimana hal ini bisa bekerja?



Gambar V.24 Lintasan USV

Autopilot dirancang untuk mengarahkan USV ke titik tuju dan memanipulasi *rudder* sedemikian rupa untuk mengurangi perbedaan antara lintasan yang diinginkan dan lintasan aktual yang ditempuh USV. Dalam hal ini, autopilot bergantung padasistem kontrol. Pada

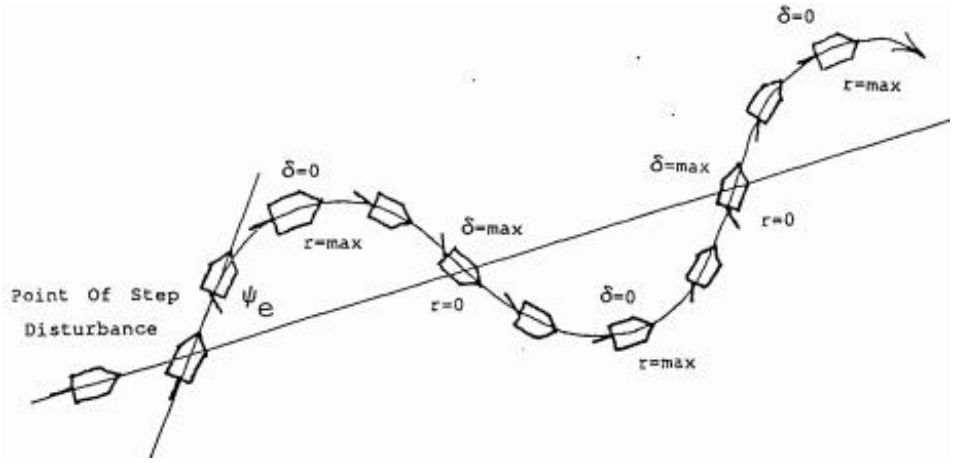


sistem kontrol dikenal yang namanya PID (*Proportional, Integral*, dan *Derivative*).Gambar V.25 akan menjelaskan tentang diagram kerja PID.

Gambar V.25 Blok Diagram Sistem Kontrol

- *Proportional Control*

Faktor-faktor pengganggu menyebabkan arah kapal berbelok hingga terjadi ketidaksesuaian antaralintasan yang diinginkan dan aktual.Pada saat itu terjadi terdeteksi*error*, USV akan melakukan koreksi dan*rudder* bereaksi untuk membawa perahu kembali ke lintasan yang diinginkan.Namun, saat menuju lintasan yang diinginkan, akan terjadi *overshot* dan terjadi *error* lagi kearah sebaliknya. Kemudi USV akan melakukan koreksi lagi dan *rudder* bereaksi untuk membawa perahu kembali ke lintasan yang diinginkan Hal ini akan terjadi terus menerus dan menyebabkan biaya bahan bakar, waktu tambahan, dan menurunkanefisiensi.

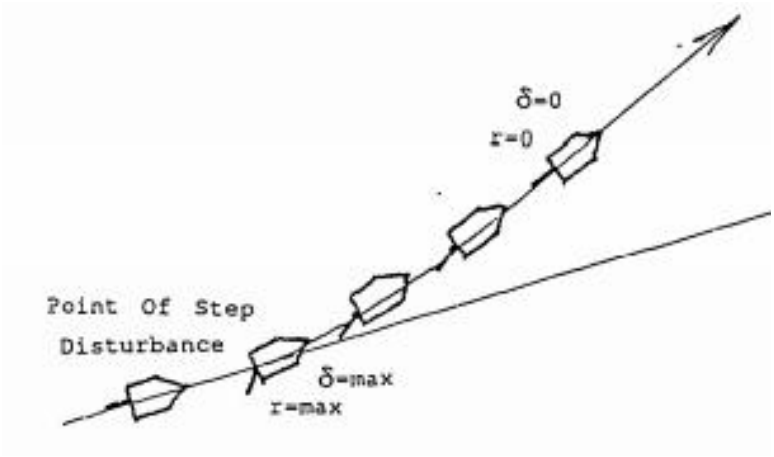


Gambar V.26 Gerakan *Proportional*
(Sumber : Browning, 1990)

- *Derivative Control*

Kontrol ini hanya terjadi ketika USV sedang berjalan dan mengalami *error.* Kontrol ini akan menghitung selisih *error* yang terjadi sekarang dengan *error* yang terjadi

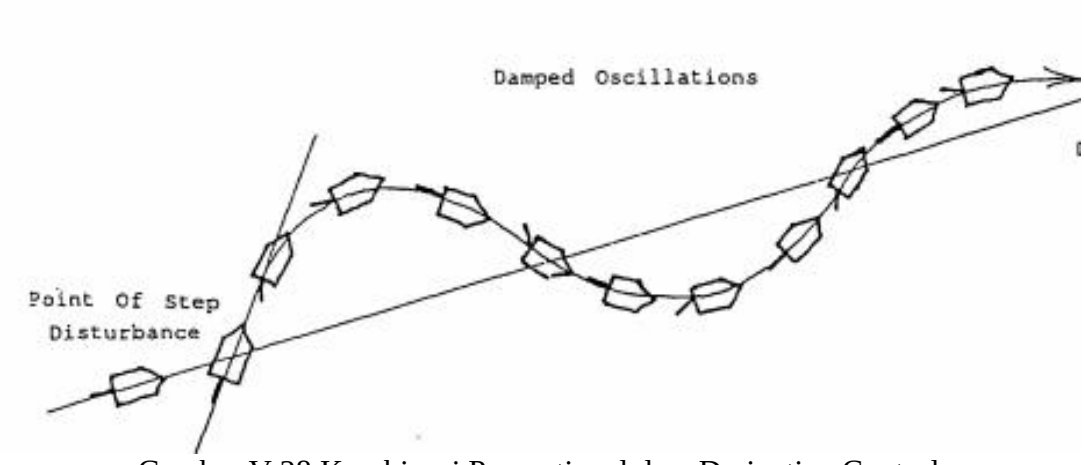
sebelumnya. Ketika pergerakan USV mengalami *overshoot* dari lintasannya, kontrol ini memerintahkan *rudder*bertindak berlawanan arahdan menyediakan "*counter rudder*" untuk menurunkan lajupergerakan USV.



Gambar V.27 Gerakan *Derivative*
(Sumber : Browning, 1990)

- Kombinasi*Proportional*dan *Derivative Control*

Pada saat itu terjadi terdeteksi*error*, kontrol *Proportional* akan melakukan koreksi dan*rudder* akan bereaksi untuk membawa perahu kembali ke lintasan yang diinginkan.Namun, saat menuju lintasan yang diinginkan, akan terjadi *overshot* dan terjadi *error* lagi kearah sebaliknya. Ketika pergerakan USV mengalami *overshoot* dari lintasannya, kontrol *Derivative* memerintahkan *rudder*bertindak berlawanan arahdan menyediakan "*counter rudder*" untuk menurunkan lajupergerakan USV.



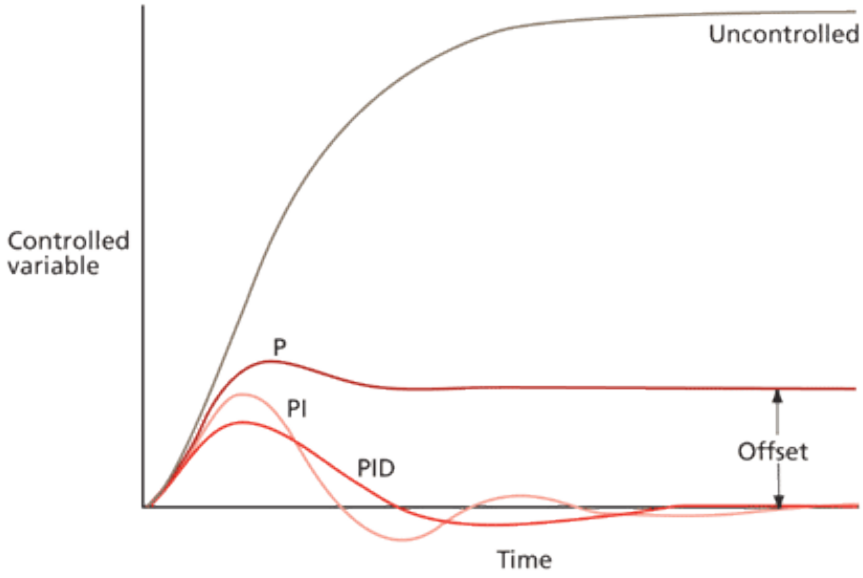
Gambar V.28 Kombinasi Proportional dan Derivative Control
(Sumber : Browning, 1990)

- *Integral Control*

Kontrol *Integral* berfungsi menghilangkan *Error Steady State* namun dapat menimbulkan ketidakstabilan (karena menambah orde sistem). Selain itu kontrol integral memiliki respon yang lebih lama dibandingkan kontrol proporsional.

- Kombinasi PID

Ada kemungkinan gangguan terus terjadi, misalny, angin dan gelombang. Momen yang dihasilkan oleh posisi *rudder* yang dihitung oleh pengontrol PD. Tingkat *error* tidak dapat dihilangkan tetapi dapat dikurangi. Caranya dengan penambahan *integral*secara efektif selama proses*looping*. Proses diulang-ulang dengan kecepatan pengambilan data per milisekon sehingga seolah-olah data diolah bersamaan, padahal pengolahannya dilakukan secara bergantian.

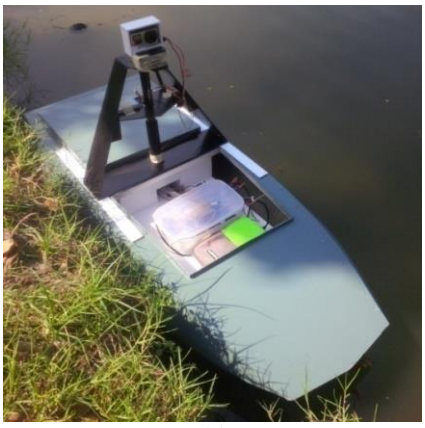


Gambar V.29 Respon untuk Perubahan Kontrol Akibat Gangguan pada P, PI, dan PID

V.5. Pengujian Fungsi Sistem Kamera *Monitoring*

Pengujian sistem kamera *monitoring* adalah pengujian yang dilakukan untuk mengetahui fungsional sistem kamera*monitoring*yang terdapat pada purwarupa USV. Pengujian sistem kamera *monitoring* purwarupa USV dilakukan setelah lambung platform, komponen sistem propulsi, dan komponen sistem autopilot purwarupa USV sudah selesai diuji. Pengujian yang dilakukan yaitu akses kamera *action* GoPro Hero 4 purwarupa USV dari GCS, dan dilanjutkan uji coba di danau untuk fungsional kamera *action*. Pengujian dilakukan di danau 8 ITS pada tanggal 20 Juni 2017.

Gambar V.30Akses Kamera *Action* GoPro Hero 4



(a)

(b)

Gambar V.1 (a) dan (b) Uji Fungsional Kamera *Monitoring* di Air

Tabel V.11 Hasil Pengujian Sistem Propulsi

No.	Parameter Pengujian	Hasil	Keterangan
1	Akses Kamera *Action* Melalui GCS	Baik	Kamera *action* dapat diakses melalui GCS
2	Fungsional Kamera *Action* Ketika USV Bergerak di Air	Baik	Kamera *action* berfungsi dengan baik ketika USV Bergerak di Air

Halaman ini sengaja dikosongkan.

BAB VI
KESIMPULAN DAN SARAN

VI.1. Kesimpulan

Setelah dilakukan pembuatan dan percobaan, maka kesimpulan dari Tugas Akhir ini adalah sebagai berikut:

1. Penentuan ukuran utama USV berdasarkan skala ukuran mesin induk yang tersedia untuk tes model. Skala yang digunakan adalah 1 : 7.Dari data tersebut, kemudian didapat ukuran utama USVyaitu:
   - Panjang (LWL) : 98.6 cm
   - Tinggi (H) : 16.5 cm
   - Lebar (B) : 50 cm
   - Sarat (T) : 7 cm
   - Displacement : 9.78 kg
   - Kecepatan (Vm) : 7.56 knot = 3.889 m/s
2. Hasil pengujian untuk platform purwarupa didapat bahwa pengujian untuk kebocoran, daya apung, dan kestabilan dinyatakan baik dengan tidak adanya kebocoran, purwarupa dapat mengapung dengan baik, dan stabil pada saat sesuai garis sarat.
3. Hasil pengujian untuk sistem propulsi didapat bahwa pengendalian fungsi mode manual, motor *brushless, rudder,* dan *propeller* dapat bekerja dengan baik. Pada pengujian kecepatan, untuk kecepatan maksimal yang dapat dicapai adalah 4.06 m/s atau 7.892 knot.
4. Hasil pengujian konektivitas aplikasi *Mission Planner*dan ArduPilot Mega 2.8 didapat bahwa akses APM 2.8 dengan *software Mission Planner*dan komunikasi antar kedua perangkat dinyatakan berhasil. Pengujian dilakukan sebanyak 20 kali dengan masing-masing tingkat berhasil mencapai 15 dan 18 kali, sementara tingkat kegagalan mencapai 5 dan 2 kali.
5. Hasil pengujian untuk sensor GPS dan kompas dinyatakan berhasil. Pengujian dilakukan sebanyak 20 kali dengan masing-masing tingkat berhasil mencapai 7 dan 9 kali, sementara tingkat kegagalan mencapai 13 dan 11 kali. Tetapi, setiap kali mengalami kegagalan, maka sensor dikalibrasi lagi sehingga mendekati tingkat akurat sehingga pengujian dapat dinyatakan berhasil.

6. Hasil pengujian untuk sistem autopilot mode manual dinyatakan berhasil. Parameter pengujian yaitu pergerakan USV, koordinat posisi USV, arah kompas USV, kecepatan USV, dan maneuver USV, semua dapat dipantau melalui *Ground Control Station* (GCS)
7. Hasil pengujian untuk sistem autopilot mode autopilot dinyatakan berhasil. Dari 4 tipe lintasan, USV bergerak tepat menuju *waypoint* walaupun lintasan yang ditempuh tidak sesuai lintasan yang diinginkan
8. Hasil pengujian untuk sistem kamera *monitoring* didapat bahwa akses kamera *action* melalui GCS dan fungsional kamera *action* ketika USV brgerak di air dinyatakan baik. Kamera *action* dapat diakses melalui GCS dan GCS dapat menampilkan citra gambar video dari kamera *monitoring* secara *real time*.

VI.2. Saran

1. Perlu adanya penelitian lebih lanjut untuk penggunaan *Personal Computer* (PC) sebagai pengganti mokrokontroler untuk mengembangkan USV dalam ukuran yang sebenarnya.
2. Perlu adanya penelitian lebih lanjut untuk menambahkan sensor lain seperti LIDAR ,RADAR,*Image processing*, dan lainnya untuk menambah kehandalan dari sistem navigasi yang telah dibahas sebelumnya.

DAFTAR PUSTAKA

Browning, A W. (1990). *A Mathematical Model to Simulate Small Boat Behaviour*. Dorset : Bournemouth Polytechnic.

Corrigam, Fintan. (2017). *FPV Goggles For Drones*. Retrieved June 04, 2017, from DroneZon : DroneZon.com

Didit. (2013, April 01). *Pengertian Robot.* Retrieved May 04, 2017, from Didit Note : http://diditnote.blogspot.co.id/2013/04/pengertian-robot.html

Hutajulu, O. P. (2010). *Performansi Tracking H-Inf Menggunakan Model Fuzzy Takagi-Sugeno pada Sistem Pendulum Terbalik.* Surabaya : ITS.

Goetz, Dietrich. (2013). *Communication, Navigation, and Control of anAutonomous Mobile Robot for Arctic and Antaracrtic Science.* Darthmount :Thayer School Of EngineeringDarthmouth college.

Hardianto, Dwiko. (2017). Tugas Akhir. *Pembuatan Konsep Desain Unmanned Surface Vehicle (USV) untuk Monitoring Wilayah Perairan Indonesia*. Surabaya : ITS.

Jono, G. (2017, April 14). *Teori Dasar GSM.* Retrieved May 04, 2017, from Jono : https://groupjono.wordpress.com/2012/05/29/teori-dasar-gsm/

Lowrie, William. (2007). *Fundamentals of Geophysics*. London: Cambridge University.

Musbikhin. (2012, May 29). *Baterai Li-Po (Lithium-Polimer).* Retrieved June 12, 2017, from Musbikhin : http://www.musbikhin.com/baterai-li-po-lithium-polimer

Nurisma, F.N, Basuki, Panggih. (2013). Tugas Akhir. *Purwarupa Robot Kapal Selam Menggunakan KontrolPDBerbasisMikrokontroller*. Yogyakarta: Universitas GajahMada.

Oborne, M. (2012). *APM Planner 1.1.59.* United Kingdom : Unmanned Tech.

Ogata K.(1997). *Modern Control Engineering.* New Jersey : Prentice-Hall.

Parkinson, B.W. (1996), *Global Positioning System: Theory and Applications.* Washington, D.C : American Institute of Aeronautics and Astronautics.

Putri, Anindita. (2016, 10 12). *Pengertian Sistem Kontrol/Sistem Kendali.* Retrieved July 05, 2017, from PT Dipta Kencana Teknologi : http://www.diptakencana.co.id/pengertian-sistem-kontrol-sistem-kendali/

Ramadhan, Fajar. (2017). Tugas Akhir. *Pembuatan Detail Desain Unmanned Surface Vehicle (USV) untuk Monitoring Wilayah Perairan Indonesia*. Surabaya : ITS.

Sulistiyanti, S.R dan Komarudin, M. (2017). Jurnal. *Sistem Navigasi pada Unmanned Surface Vehicle untuk Pemantauan Daerah Perairan.* Lampung : Universitas Lampung.

Tim Wescott. (2000). *PID without a PhD.* India : EE Times.

Turley, Jim (2002). *The Two Percent Solution Mikrokontroler.* New Jersey : Prentice-Hall.

Lim, C C. (1980). *Autopilot Design for Ship Control.* Loughboroug : Loughboroug University of Technology.

Veselý, V., Rosinová, D. (2011). *Robust PSD Controller Design.* Slowakia : Tatranská Lomnica.

LAMPIRAN

LAMPIRAN A

Datasheet Komponen Propulsi dan Autopilot

Datasheet Motor *Brushless*

The Turnigy AquaStar Sensorless Brushless Motors offer outstanding performance for your boat at an incredible price!
Featuring a quality CNC machined billet T6 aluminum motor can, hand-wound high purity copper windings and powerful sintered neodymium magnet, you wont find a better priced motor of this caliber anywhere else! It even features a billet aluminum water cooling jacket pre-installed. The AquaStar brushless motors are a great choice for your deep-V, hydro or catamaran race boat and will be sure to leave the competition in your wake!

Features:

- CNC Machined Billet T6 Aluminum Motor Can
- High Purity Copper Windings
- 4mm Bullet Connectors Pre-installed
- Powerful Sintered Neodymium Magnet
- Precision Engineered for Maximum Energy Conversion
- Water Cooling Jacket Pre-installed

Specs:

RPM/v: **2200kv**
Poles: **4**
Max voltage: **18.5V (5S)**
Max Current: **100A**
Max Watts: **1200w**
Resistance: **0.0094ohm**
No Load Current: **2.9A**
Can Diameter: **39mm (actual motor diameter)**
Can Diameter inc. Water Jacket: **49mm**
Can Length: **69mm**
Shaft Size: **5mm**
Weight: **415g**

SKU:	9192000053	Brand:	Turnigy
Weight(g)	550.00	Length	135.00
Width:	60.00	Height:	95.00
Kv(rpm/v)	2200.00	Max Currents (A)	100.00
Power (W):	1200.00	Length B (mm):	74.00
Can Length D(mm)	69.00	Max Voltage(V)	19.00
Shaft A(mm):	5.00	Unit Weight (g):	415
Diameter C(mm)	39.00	Total Length E(mm)	94.00
Resistance (mh):	0.00		

Datasheet Motor Servo

Home / Servos & Parts / X-Large Servo 50g+ / MG945

MG945

MG945 Towerpro Digital Metal Servo 12KG High torque
MG945 is for 1/8 buggy monster and many RC model.
It is low cost of digital servo with export quality and great servo for just about any application.

Specification:

Weight: 55g
Dimension: 40.7×19.7×42.9mm
Stall torque: 10kg/cm (4.8v); 12kg/cm (6v)
Operating speed: 0.23sec/60degree (4.8v); 0.2sec/60degree (6.0v)
Operating voltage: 4.8 ~ 6.6v
Temperature range: 0- 55deg
Gear Type: Metal gear
Dead band width: 1us
Power Supply: Through External Adapter
servo wire length: 32cm
Servo Plug: JR (Fits JR and Futaba)
servo arms &screws included and fit with Futaba servo arm
It's universal "S" type connector that fits most receivers, including Futaba, JR, Hitec ,GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum.
CE &RoHS approved

We have upgraded our servo gear set and shaft to aluminum 6061-T6.
It is stronger and lighter than copper.

Categories: Servos & Parts, X-Large Servo 50g+

Datasheet ESC

Specs:
Constant Current: **90A**
Burst Current: **120A**
Battery: **2-6S Lipoly**
BEC: **5.5v / 4A**
Motor Type: **Sensorless Brushless**
Cooling: **Watercooled**
Size: **87 x 38 x 22mm**
Weight: **105g**
Programming Functions:
Running Mode: **Forward with brake / Forward and reverse with brake / Forward and reverse**
Acceleration: **6% / 9% / 12%**
Low Voltage Protection: **None / 2.6V / 2.8V / 3.0V / 3.2V / 3.4V**
Start Mode (Punch): **Level 1 / Level 2 / Level 3 / Level 4 / Level 5 / Level 6 / Level 7 / Level 8 / Level 9**
Reverse Force: **25% / 50% / 75% / 100%**
Timing: **0° / 5° / 10° / 15° / 20° / 25° / 30° / Automatic**

SKU:	9261000012	Brand:	HobbyKing
Weight(g)	137.00	Length	90.00
Width:	25.00	Height:	60.00

Datasheet ArduPilot 2.8

Description

ARDUCOPTER AMP 2.8 Mega comes from the open source project ArduPilot, it is 100% compatible with ArduPilot Mega 2.8.

The APM 2.8 is a complete open source autopilot system and the bestselling technology that won the prestigious 2012 Outback Challenge UAV competition. It allows the user to turn any fixed, rotary wing or multirotor vehicle (even cars and boats) into a fully autonomous vehicle; capable of performing programmed GPS missions with waypoints.

This revision of the board has no onboard compass, which is designed for vehicles (especially multicopters and rovers) where the compass should be placed as far from power and motor sources as possible to avoid magnetic interference. APM 2.8 is designed to be used with GPS with Compass, so that the GPS/Compass unit can be mounted further from noise sources.

APM 2.8 requires a GPS unit with an onboard compass or an external compass module for full autonomy. If you are using APM 2.8 with a GPS module that does not have a compass sensor, you must use a stand-alone external compass.

Features

- Arduino Compatible!
- Includes 3-axis gyro, accelerometer and high-performance barometer
- Onboard 4 MegaByte Dataflash chip for automatic datalogging
- One of the first open-source autopilot systems to use Invensense's 6 DoF Accelerometer/Gyro MPU-6000.
- Barometric pressure sensor upgraded to MS5611-01BA03, from Measurement Specialties.
- Atmel's ATMEGA2560 and ATMEGA32U-2 chips for processing and usb functions respectively.

Datasheet GPS UBLOX M8N

Parameter	Specification					
Receiver type	72-channel u-blox M8 engine GPS L1C/A, SBAS L1C/A, QZSS L1C/A, QZSS L1 SAIF, GLONASS L1OF, BeiDou B1I, Galileo E1B/C					
Accuracy of time pulse signal	RMS 99%	30 ns 60 ns				
Frequency of time pulse signal		0.25 Hz...10 MHz (configurable)				
Operational limits [1]	Dynamics	≤ 4 g				
	Altitude	50,000 m				
	Velocity	500 m/s				
Velocity accuracy [2]		0.05m/s				
Heading accuracy [2]		0.3 degrees				
GNSS		**GPS & GLONASS**	**GPS**	**GLONASS**	**BeiDou**	**Galileo**
Horizontal position accuracy [3]		2.5 m	2.5 m	4 m	3 m	TBC[4]
NEO-M8N/Q						
Max navigation update rate	NEO-M8N	5 Hz	10 Hz	10 Hz	10 Hz	10 Hz
	NEO-M8Q	10 Hz	18 Hz	18 Hz	18 Hz	18 Hz
Time-To-First-Fix [5]	Cold start	26 s	29 s	30 s	34 s	45 s
	Hot start	1 s	1 s	1 s	1 s	1 s
	Aided starts [6]	2 s	2 s	2 s	3 s	7 s
Sensitivity[7]	Tracking & Navigation	–167 dBm	–166 dBm	-166 dBm	-160 dBm	-159 dBm
	Reacquisition	–160 dBm	–160 dBm	-156 dBm	-157 dBm	-153 dBm
	Cold start	–148 dBm	–148 dBm	-145 dBm	-143 dBm	-138 dBm
	Hot start	–157 dBm	–157 dBm	-156 dBm	-155 dBm	-151 dBm

LAMPIRAN B

Algoritma Program pada *Library* APM 2.8

***Library* APM 2.8 untuk Servo**

```
/*
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
/*
 *   AP_ServoRelayEvents - handle servo and relay MAVLink events
 */


#include <AP_HAL/AP_HAL.h>
#include <AP_Common/AP_Common.h>
#include "AP_ServoRelayEvents.h"
#include <RC_Channel/RC_Channel.h>
#include <SRV_Channel/SRV_Channel.h>

extern const AP_HAL::HAL& hal;

bool AP_ServoRelayEvents::do_set_servo(uint8_t _channel, uint16_t pwm)
{
    if (!(mask & 1U<<(_channel-1))) {
        // not allowed
        return false;
    }
    if (type == EVENT_TYPE_SERVO &&
        channel == _channel) {
        // cancel previous repeat
        repeat = 0;
    }
    SRV_Channel *c = SRV_Channels::srv_channel(_channel-1);
    if (c == nullptr) {
        return false;
    }
    c->set_output_pwm(pwm);
    return true;
}

bool AP_ServoRelayEvents::do_set_relay(uint8_t relay_num, uint8_t state)
```

```
{
    if (!relay.enabled(relay_num)) {
        return false;
    }
    if (type == EVENT_TYPE_RELAY &&
        channel == relay_num) {
        // cancel previous repeat
        repeat = 0;
    }
    if (state == 1) {
        relay.on(relay_num);
    } else if (state == 0) {
        relay.off(relay_num);
    } else {
        relay.toggle(relay_num);
    }
    return true;
}

bool AP_ServoRelayEvents::do_repeat_servo(uint8_t _channel, uint16_t _servo_value,
                                          int16_t _repeat, uint16_t _delay_ms)
{
    if (!(mask & 1U<<(_channel-1))) {
        // not allowed
        return false;
    }
    channel = _channel;
    type = EVENT_TYPE_SERVO;

    start_time_ms  = 0;
    delay_ms    = _delay_ms / 2;
    repeat      = _repeat * 2;
    servo_value = _servo_value;
    update_events();
    return true;
}

bool AP_ServoRelayEvents::do_repeat_relay(uint8_t relay_num, int16_t _repeat, uint32_t
_delay_ms)
{
    if (!relay.enabled(relay_num)) {
        return false;
    }
    type = EVENT_TYPE_RELAY;
    channel = relay_num;
    start_time_ms  = 0;
    delay_ms       = _delay_ms/2; // half cycle time
    repeat         = _repeat*2;  // number of full cycles
    update_events();
```

```
    return true;
}


/*
  update state for MAV_CMD_DO_REPEAT_SERVO and
MAV_CMD_DO_REPEAT_RELAY
*/
void AP_ServoRelayEvents::update_events(void)
{
    if (repeat == 0 || (AP_HAL::millis() - start_time_ms) < delay_ms) {
        return;
    }

    start_time_ms = AP_HAL::millis();

    switch (type) {
    case EVENT_TYPE_SERVO: {
        SRV_Channel *c = SRV_Channels::srv_channel(channel-1);
        if (c != nullptr) {
            if (repeat & 1) {
                c->set_output_pwm(c->get_trim());
            } else {
                c->set_output_pwm(servo_value);
            }
        }
        break;
    }

    case EVENT_TYPE_RELAY:
        relay.toggle(channel);
        break;
    }

    if (repeat > 0) {
        repeat--;
    } else {
        // toggle bottom bit so servos flip in value
        repeat ^= 1;
    }
}
```

***Library* APM 2.8 untuk Servo**

```
/*
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

//
//  u-blox GPS driver for ArduPilot
//         Origin code by Michael Smith, Jordi Munoz and Jose Julio, DIYDrones.com
//  Substantially rewritten for new GPS driver structure by Andrew Tridgell
//
#include "AP_GPS.h"
#include "AP_GPS_UBLOX.h"
#include <AP_HAL/Util.h>
#include <DataFlash/DataFlash.h>
#include <GCS_MAVLink/GCS.h>

#if CONFIG_HAL_BOARD_SUBTYPE == HAL_BOARD_SUBTYPE_LINUX_NAVIO || \
    CONFIG_HAL_BOARD_SUBTYPE == HAL_BOARD_SUBTYPE_LINUX_BH
    #define UBLOX_SPEED_CHANGE  1
#else
    #define UBLOX_SPEED_CHANGE 0
#endif


#define UBLOX_DEBUGGING 0
#define UBLOX_FAKE_3DLOCK 0

extern const AP_HAL::HAL& hal;

#if UBLOX_DEBUGGING
 # define Debug(fmt, args ...)  do {hal.console->printf("%s:%d: " fmt "\n", __FUNCTION__,
__LINE__, ## args); hal.scheduler->delay(1); } while(0)
#else
 # define Debug(fmt, args ...)
#endif

AP_GPS_UBLOX::AP_GPS_UBLOX(AP_GPS &_gps, AP_GPS::GPS_State &_state,
AP_HAL::UARTDriver *_port) :
```

```
    AP_GPS_Backend(_gps, _state, _port),
    _step(0),
    _msg_id(0),
    _payload_length(0),
    _payload_counter(0),
    _class(0),
    _cfg_saved(false),
    _last_cfg_sent_time(0),
    _num_cfg_save_tries(0),
    _last_config_time(0),
    _delay_time(0),
    _next_message(STEP_PVT),
    _ublox_port(255),
    _have_version(false),
    _unconfigured_messages(CONFIG_ALL),
    _hardware_generation(UBLOX_UNKNOWN_HARDWARE_GENERATION),
    _new_position(0),
    _new_speed(0),
    _disable_counter(0),
    next_fix(AP_GPS::NO_FIX),
    _cfg_needs_save(false),
    noReceivedHdop(true),
    havePvtMsg(false)
{
    // stop any config strings that are pending
    gps.send_blob_start(state.instance, nullptr, 0);

    // start the process of updating the GPS rates
    _request_next_config();
}

void
AP_GPS_UBLOX::_request_next_config(void)
{
    // don't request config if we shouldn't configure the GPS
    if (gps._auto_config == AP_GPS::GPS_AUTO_CONFIG_DISABLE) {
        return;
    }

    // Ensure there is enough space for the largest possible outgoing message
    if (port->txspace() < (int16_t)(sizeof(struct ubx_header)+sizeof(struct
ubx_cfg_nav_rate)+2)) {
        // not enough space - do it next time
        return;
    }

   Debug("Unconfigured messages: %d Current message: %d\n", _unconfigured_messages,
_next_message);
```

```
// check AP_GPS_UBLOX.h for the enum that controls the order.
// This switch statement isn't maintained against the enum in order to reduce code churn
 switch (_next_message++) {
 case STEP_PVT:
    if(!_request_message_rate(CLASS_NAV, MSG_PVT)) {
       _next_message--;
    }
    break;
 case STEP_PORT:
    _request_port();
    break;
 case STEP_POLL_SVINFO:
    // not required once we know what generation we are on
    if(_hardware_generation == UBLOX_UNKNOWN_HARDWARE_GENERATION) {
       if (!_send_message(CLASS_NAV, MSG_NAV_SVINFO, 0, 0)) {
          _next_message--;
       }
    }
    break;
 case STEP_POLL_SBAS:
    if (gps._sbas_mode != 2) {
       _send_message(CLASS_CFG, MSG_CFG_SBAS, nullptr, 0);
    } else {
       _unconfigured_messages &= ~CONFIG_SBAS;
    }
    break;
 case STEP_POLL_NAV:
    if (!_send_message(CLASS_CFG, MSG_CFG_NAV_SETTINGS, nullptr, 0)) {
       _next_message--;
    }
    break;
 case STEP_POLL_GNSS:
    if (!_send_message(CLASS_CFG, MSG_CFG_GNSS, nullptr, 0)) {
       _next_message--;
    }
    break;
 case STEP_NAV_RATE:
    if (!_send_message(CLASS_CFG, MSG_CFG_RATE, nullptr, 0)) {
       _next_message--;
    }
    break;
 case STEP_POSLLH:
    if(!_request_message_rate(CLASS_NAV, MSG_POSLLH)) {
       _next_message--;
    }
    break;
 case STEP_STATUS:
    if(!_request_message_rate(CLASS_NAV, MSG_STATUS)) {
       _next_message--;
```

```
        }
        break;
    case STEP_SOL:
        if(!_request_message_rate(CLASS_NAV, MSG_SOL)) {
            _next_message--;
        }
        break;
    case STEP_VELNED:
        if(!_request_message_rate(CLASS_NAV, MSG_VELNED)) {
            _next_message--;
        }
        break;
    case STEP_DOP:
       if(! _request_message_rate(CLASS_NAV, MSG_DOP)) {
            _next_message--;
        }
        break;
    case STEP_MON_HW:
        if(!_request_message_rate(CLASS_MON, MSG_MON_HW)) {
            _next_message--;
        }
        break;
    case STEP_MON_HW2:
        if(!_request_message_rate(CLASS_MON, MSG_MON_HW2)) {
            _next_message--;
        }
        break;
    case STEP_RAW:
#if UBLOX_RXM_RAW_LOGGING
        if(gps._raw_data == 0) {
            _unconfigured_messages &= ~CONFIG_RATE_RAW;
        } else if(!_request_message_rate(CLASS_RXM, MSG_RXM_RAW)) {
            _next_message--;
        }
#else
        _unconfigured_messages & = ~CONFIG_RATE_RAW;
#endif
        break;
    case STEP_RAWX:
#if UBLOX_RXM_RAW_LOGGING
        if(gps._raw_data == 0) {
            _unconfigured_messages &= ~CONFIG_RATE_RAW;
        } else if(!_request_message_rate(CLASS_RXM, MSG_RXM_RAWX)) {
            _next_message--;
        }
#else
        _unconfigured_messages & = ~CONFIG_RATE_RAW;
#endif
        break;
```

```
    case STEP_VERSION:
        if(!_have_version && !hal.util->get_soft_armed()) {
            _request_version();
        } else {
            _unconfigured_messages &= ~CONFIG_VERSION;
        }
        // no need to send the initial rates, move to checking only
        _next_message = STEP_PVT;
        break;
    default:
        // this case should never be reached, do a full reset if it is hit
        _next_message = STEP_PVT;
        break;
    }
}

void
AP_GPS_UBLOX::_verify_rate(uint8_t msg_class, uint8_t msg_id, uint8_t rate) {
    uint8_t desired_rate;

    switch(msg_class) {
    case CLASS_NAV:
        switch(msg_id) {
        case MSG_POSLLH:
            desired_rate = havePvtMsg ? 0 : RATE_POSLLH;
            if(rate == desired_rate) {
                _unconfigured_messages &= ~CONFIG_RATE_POSLLH;
            } else {
                _configure_message_rate(msg_class, msg_id, desired_rate);
                _unconfigured_messages |= CONFIG_RATE_POSLLH;
                _cfg_needs_save = true;
            }
            break;
        case MSG_STATUS:
            desired_rate = havePvtMsg ? 0 : RATE_STATUS;
            if(rate == desired_rate) {
                _unconfigured_messages &= ~CONFIG_RATE_STATUS;
            } else {
                _configure_message_rate(msg_class, msg_id, desired_rate);
                _unconfigured_messages |= CONFIG_RATE_STATUS;
                _cfg_needs_save = true;
            }
            break;
        case MSG_SOL:
            if(rate == RATE_SOL) {
                _unconfigured_messages &= ~CONFIG_RATE_SOL;
            } else {
                _configure_message_rate(msg_class, msg_id, RATE_SOL);
                _unconfigured_messages |= CONFIG_RATE_SOL;
```

```
            _cfg_needs_save = true;
        }
        break;
    case MSG_PVT:
        if(rate == RATE_PVT) {
            _unconfigured_messages &= ~CONFIG_RATE_PVT;
        } else {
            _configure_message_rate(msg_class, msg_id, RATE_PVT);
            _unconfigured_messages |= CONFIG_RATE_PVT;
            _cfg_needs_save = true;
        }
        break;
    case MSG_VELNED:
        desired_rate = havePvtMsg ? 0 : RATE_VELNED;
        if(rate == desired_rate) {
            _unconfigured_messages &= ~CONFIG_RATE_VELNED;
        } else {
            _configure_message_rate(msg_class, msg_id, desired_rate);
            _unconfigured_messages |= CONFIG_RATE_VELNED;
            _cfg_needs_save = true;
        }
        break;
    case MSG_DOP:
        if(rate == RATE_DOP) {
            _unconfigured_messages &= ~CONFIG_RATE_DOP;
        } else {
            _configure_message_rate(msg_class, msg_id, RATE_DOP);
            _unconfigured_messages |= CONFIG_RATE_DOP;
            _cfg_needs_save = true;
        }
        break;
    }
    break;
case CLASS_MON:
    switch(msg_id) {
    case MSG_MON_HW:
        if(rate == RATE_HW) {
            _unconfigured_messages &= ~CONFIG_RATE_MON_HW;
        } else {
            _configure_message_rate(msg_class, msg_id, RATE_HW);
            _unconfigured_messages |= CONFIG_RATE_MON_HW;
            _cfg_needs_save = true;
        }
        break;
    case MSG_MON_HW2:
        if(rate == RATE_HW2) {
            _unconfigured_messages &= ~CONFIG_RATE_MON_HW2;
        } else {
            _configure_message_rate(msg_class, msg_id, RATE_HW2);
```

```
            _unconfigured_messages |= CONFIG_RATE_MON_HW2;
            _cfg_needs_save = true;
         }
         break;
      }
      break;
#if UBLOX_RXM_RAW_LOGGING
   case CLASS_RXM:
      switch(msg_id) {
      case MSG_RXM_RAW:
         if(rate == gps._raw_data) {
            _unconfigured_messages &= ~CONFIG_RATE_RAW;
         } else {
            _configure_message_rate(msg_class, msg_id, gps._raw_data);
            _unconfigured_messages |= CONFIG_RATE_RAW;
            _cfg_needs_save = true;
         }
         break;
      case MSG_RXM_RAWX:
         if(rate == gps._raw_data) {
            _unconfigured_messages &= ~CONFIG_RATE_RAW;
         } else {
            _configure_message_rate(msg_class, msg_id, gps._raw_data);
            _unconfigured_messages |= CONFIG_RATE_RAW;
            _cfg_needs_save = true;
         }
         break;
      }
      break;
#endif // UBLOX_RXM_RAW_LOGGING
   }
}

// Requests the ublox driver to identify what port we are using to communicate
void
AP_GPS_UBLOX::_request_port(void)
{
   if (port->txspace() < (int16_t)(sizeof(struct ubx_header)+2)) {
      // not enough space - do it next time
      return;
   }
   _send_message(CLASS_CFG, MSG_CFG_PRT, nullptr, 0);
}
   // Ensure there is enough space for the largest possible outgoing message
// Process bytes available from the stream
//
// The stream is assumed to contain only messages we recognise.  If it
// contains other messages, and those messages contain the preamble
// bytes, it is possible for this code to fail to synchronise to the
```

```
// stream immediately.  Without buffering the entire message and
// re-processing it from the top, this is unavoidable. The parser
// attempts to avoid this when possible.
//
bool
AP_GPS_UBLOX::read(void)
{
    uint8_t data;
    int16_t numc;
    bool parsed = false;
    uint32_t millis_now = AP_HAL::millis();

    // walk through the gps configuration at 1 message per second
    if (millis_now - _last_config_time >= _delay_time) {
        _request_next_config();
        _last_config_time = millis_now;
        if (_unconfigured_messages) { // send the updates faster until fully configured
            if (!havePvtMsg && (_unconfigured_messages & CONFIG_REQUIRED_INITIAL))
{
                _delay_time = 300;
            } else {
                _delay_time = 750;
            }
        } else {
            _delay_time = 2000;
        }
    }

    if(!_unconfigured_messages && gps._save_config && !_cfg_saved &&
       _num_cfg_save_tries < 5 && (millis_now - _last_cfg_sent_time) > 5000 &&
       !hal.util->get_soft_armed()) {
        //save the configuration sent until now
        if (gps._save_config == 1 ||
            (gps._save_config == 2 && _cfg_needs_save)) {
            _save_cfg();
        }
    }

    numc = port->available();
    for (int16_t i = 0; i < numc; i++) {        // Process bytes received

        // read the next byte
        data = port->read();

            reset:
        switch(_step) {

        // Message preamble detection
        //
```

```
// If we fail to match any of the expected bytes, we reset
// the state machine and re-consider the failed byte as
// the first byte of the preamble.  This improves our
// chances of recovering from a mismatch and makes it less
// likely that we will be fooled by the preamble appearing
// as data in some other message.
//
case 1:
   if (PREAMBLE2 == data) {
      _step++;
      break;
   }
   _step = 0;
   Debug("reset %u", __LINE__);
   /* no break */
case 0:
   if(PREAMBLE1 == data)
      _step++;
   break;

// Message header processing
//
// We sniff the class and message ID to decide whether we
// are going to gather the message bytes or just discard
// them.
//
// We always collect the length so that we can avoid being
// fooled by preamble bytes in messages.
//
case 2:
   _step++;
   _class = data;
   _ck_b = _ck_a = data;                       // reset the checksum accumulators
   break;
case 3:
   _step++;
   _ck_b += (_ck_a += data);                   // checksum byte
   _msg_id = data;
   break;
case 4:
   _step++;
   _ck_b += (_ck_a += data);                   // checksum byte
   _payload_length = data;                     // payload length low byte
   break;
case 5:
   _step++;
   _ck_b += (_ck_a += data);                   // checksum byte

   _payload_length += (uint16_t)(data<<8);
```

```
            if (_payload_length > sizeof(_buffer)) {
                Debug("large payload %u", (unsigned)_payload_length);
                // assume any payload bigger then what we know about is noise
                _payload_length = 0;
                _step = 0;
                                goto reset;
            }
            _payload_counter = 0;                           // prepare to receive payload
            break;

        // Receive message data
        //
        case 6:
            _ck_b += (_ck_a += data);                   // checksum byte
            if (_payload_counter < sizeof(_buffer)) {
                _buffer[_payload_counter] = data;
            }
            if (++_payload_counter == _payload_length)
                _step++;
            break;

        // Checksum and message processing
        //
        case 7:
            _step++;
            if (_ck_a != data) {
                Debug("bad cka %x should be %x", data, _ck_a);
                _step = 0;
                                goto reset;
            }
            break;
        case 8:
            _step = 0;
            if (_ck_b != data) {
                Debug("bad ckb %x should be %x", data, _ck_b);
                break;                                      // bad checksum
            }

            if (_parse_gps()) {
                parsed = true;
            }
            break;
        }
    }
    return parsed;
}

// Private Methods //////////////////////////////////////////////////////////////
void AP_GPS_UBLOX::log_mon_hw(void)
```

```
{
    if (!should_df_log()) {
        return;
    }
    struct log_Ubx1 pkt = {
        LOG_PACKET_HEADER_INIT(_ubx_msg_log_index(LOG_GPS_UBX1_MSG)),
        time_us    : AP_HAL::micros64(),
        instance   : state.instance,
        noisePerMS : _buffer.mon_hw_60.noisePerMS,
        jamInd     : _buffer.mon_hw_60.jamInd,
        aPower     : _buffer.mon_hw_60.aPower,
        agcCnt     : _buffer.mon_hw_60.agcCnt,
    };
    if (_payload_length == 68) {
        pkt.noisePerMS = _buffer.mon_hw_68.noisePerMS;
        pkt.jamInd     = _buffer.mon_hw_68.jamInd;
        pkt.aPower     = _buffer.mon_hw_68.aPower;
        pkt.agcCnt     = _buffer.mon_hw_68.agcCnt;
    }
    DataFlash_Class::instance()->WriteBlock(&pkt, sizeof(pkt));
}

void AP_GPS_UBLOX::log_mon_hw2(void)
{
    if (!should_df_log()) {
        return;
    }

    struct log_Ubx2 pkt = {
        LOG_PACKET_HEADER_INIT(_ubx_msg_log_index(LOG_GPS_UBX2_MSG)),
        time_us   : AP_HAL::micros64(),
        instance  : state.instance,
        ofsI      : _buffer.mon_hw2.ofsI,
        magI      : _buffer.mon_hw2.magI,
        ofsQ      : _buffer.mon_hw2.ofsQ,
        magQ      : _buffer.mon_hw2.magQ,
    };
    DataFlash_Class::instance()->WriteBlock(&pkt, sizeof(pkt));
}

#if UBLOX_RXM_RAW_LOGGING
void AP_GPS_UBLOX::log_rxm_raw(const struct ubx_rxm_raw &raw)
{
    if (!should_df_log()) {
        return;
    }

    uint64_t now = AP_HAL::micros64();
    for (uint8_t i=0; i<raw.numSV; i++) {
```

```
        struct log_GPS_RAW pkt = {
            LOG_PACKET_HEADER_INIT(LOG_GPS_RAW_MSG),
            time_us    : now,
            iTOW       : raw.iTOW,
            week       : raw.week,
            numSV      : raw.numSV,
            sv         : raw.svinfo[i].sv,
            cpMes      : raw.svinfo[i].cpMes,
            prMes      : raw.svinfo[i].prMes,
            doMes      : raw.svinfo[i].doMes,
            mesQI      : raw.svinfo[i].mesQI,
            cno        : raw.svinfo[i].cno,
            lli        : raw.svinfo[i].lli
        };
        DataFlash_Class::instance()->WriteBlock(&pkt, sizeof(pkt));
    }
}

void AP_GPS_UBLOX::log_rxm_rawx(const struct ubx_rxm_rawx &raw)
{
    if (!should_df_log()) {
        return;
    }

    uint64_t now = AP_HAL::micros64();

    struct log_GPS_RAWH header = {
        LOG_PACKET_HEADER_INIT(LOG_GPS_RAWH_MSG),
        time_us    : now,
        rcvTow     : raw.rcvTow,
        week       : raw.week,
        leapS      : raw.leapS,
        numMeas    : raw.numMeas,
        recStat    : raw.recStat
    };
    DataFlash_Class::instance()->WriteBlock(&header, sizeof(header));

    for (uint8_t i=0; i<raw.numMeas; i++) {
        struct log_GPS_RAWS pkt = {
            LOG_PACKET_HEADER_INIT(LOG_GPS_RAWS_MSG),
            time_us    : now,
            prMes      : raw.svinfo[i].prMes,
            cpMes      : raw.svinfo[i].cpMes,
            doMes      : raw.svinfo[i].doMes,
            gnssId     : raw.svinfo[i].gnssId,
            svId       : raw.svinfo[i].svId,
            freqId     : raw.svinfo[i].freqId,
            locktime   : raw.svinfo[i].locktime,
            cno        : raw.svinfo[i].cno,
```

```
            prStdev    : raw.svinfo[i].prStdev,
            cpStdev    : raw.svinfo[i].cpStdev,
            doStdev    : raw.svinfo[i].doStdev,
            trkStat    : raw.svinfo[i].trkStat
        };
        DataFlash_Class::instance()->WriteBlock(&pkt, sizeof(pkt));
    }
}
#endif // UBLOX_RXM_RAW_LOGGING

void AP_GPS_UBLOX::unexpected_message(void)
{
    Debug("Unexpected message 0x%02x 0x%02x", (unsigned)_class, (unsigned)_msg_id);
    if (++_disable_counter == 0) {
        // disable future sends of this message id, but
        // only do this every 256 messages, as some
        // message types can't be disabled and we don't
        // want to get into an ack war
        Debug("Disabling message 0x%02x 0x%02x", (unsigned)_class, (unsigned)_msg_id);
        _configure_message_rate(_class, _msg_id, 0);
    }
}

bool
AP_GPS_UBLOX::_parse_gps(void)
{
    if (_class == CLASS_ACK) {
        Debug("ACK %u", (unsigned)_msg_id);

        if(_msg_id == MSG_ACK_ACK) {
            switch(_buffer.ack.clsID) {
            case CLASS_CFG:
                switch(_buffer.ack.msgID) {
                case MSG_CFG_CFG:
                    _cfg_saved = true;
                    _cfg_needs_save = false;
                    break;
                case MSG_CFG_GNSS:
                    _unconfigured_messages &= ~CONFIG_GNSS;
                    break;
                case MSG_CFG_MSG:
                    // There is no way to know what MSG config was ack'ed, assume it was the last
                    // one requested. To verify it rerequest the last config we sent. If we miss
                    // the actual ack we will catch it next time through the poll loop, but that
                    // will be a good chunk of time later.
                    break;
                case MSG_CFG_NAV_SETTINGS:
                    _unconfigured_messages &= ~CONFIG_NAV_SETTINGS;
                    break;
```

```
            case MSG_CFG_RATE:
               // The GPS will ACK a update rate that is invalid. in order to detect this
               // only accept the rate as configured by reading the settings back and
               //  validating that they all match the target values
               break;
            case MSG_CFG_SBAS:
               _unconfigured_messages &= ~CONFIG_SBAS;
               break;
            }
            break;
         case CLASS_MON:
            switch(_buffer.ack.msgID) {
            case MSG_MON_HW:
               _unconfigured_messages &= ~CONFIG_RATE_MON_HW;
               break;
            case MSG_MON_HW2:
               _unconfigured_messages &= ~CONFIG_RATE_MON_HW2;
               break;
            }
         }
      }
      return false;
   }

   if (_class == CLASS_CFG) {
      switch(_msg_id) {
      case  MSG_CFG_NAV_SETTINGS:
            Debug("Got settings %u min_elev %d drLimit %u\n",
              (unsigned)_buffer.nav_settings.dynModel,
              (int)_buffer.nav_settings.minElev,
              (unsigned)_buffer.nav_settings.drLimit);
         _buffer.nav_settings.mask = 0;
         if (gps._navfilter != AP_GPS::GPS_ENGINE_NONE &&
            _buffer.nav_settings.dynModel != gps._navfilter) {
            // we've received the current nav settings, change the engine
            // settings and send them back
            Debug("Changing engine setting from %u to %u\n",
                  (unsigned)_buffer.nav_settings.dynModel, (unsigned)gps._navfilter);
            _buffer.nav_settings.dynModel = gps._navfilter;
            _buffer.nav_settings.mask |= 1;
         }
         if (gps._min_elevation != -100 &&
            _buffer.nav_settings.minElev != gps._min_elevation) {
            Debug("Changing min elevation to %d\n", (int)gps._min_elevation);
            _buffer.nav_settings.minElev = gps._min_elevation;
            _buffer.nav_settings.mask |= 2;
         }
         if (_buffer.nav_settings.mask != 0) {
            _send_message(CLASS_CFG, MSG_CFG_NAV_SETTINGS,
```

```
&_buffer.nav_settings,
                        sizeof(_buffer.nav_settings));
            _unconfigured_messages |= CONFIG_NAV_SETTINGS;
            _cfg_needs_save = true;
         } else {
            _unconfigured_messages &= ~CONFIG_NAV_SETTINGS;
         }
         return false;

#if UBLOX_GNSS_SETTINGS
      case MSG_CFG_GNSS:
         if (gps._gnss_mode[state.instance] != 0) {
            struct ubx_cfg_gnss start_gnss = _buffer.gnss;
            uint8_t gnssCount = 0;
            Debug("Got GNSS Settings %u %u %u %u:\n",
               (unsigned)_buffer.gnss.msgVer,
               (unsigned)_buffer.gnss.numTrkChHw,
               (unsigned)_buffer.gnss.numTrkChUse,
               (unsigned)_buffer.gnss.numConfigBlocks);
#if UBLOX_DEBUGGING
            for(int i = 0; i < _buffer.gnss.numConfigBlocks; i++) {
               Debug("  %u %u %u 0x%08x\n",
               (unsigned)_buffer.gnss.configBlock[i].gnssId,
               (unsigned)_buffer.gnss.configBlock[i].resTrkCh,
               (unsigned)_buffer.gnss.configBlock[i].maxTrkCh,
               (unsigned)_buffer.gnss.configBlock[i].flags);
            }
#endif

            for(int i = 0; i < UBLOX_MAX_GNSS_CONFIG_BLOCKS; i++) {
               if((gps._gnss_mode[state.instance] & (1 << i)) && i != GNSS_SBAS) {
                  gnssCount++;
               }
            }

            for(int i = 0; i < _buffer.gnss.numConfigBlocks; i++) {
               // Reserve an equal portion of channels for all enabled systems
               if(gps._gnss_mode[state.instance] & (1 << _buffer.gnss.configBlock[i].gnssId)) {
                  if(GNSS_SBAS !=_buffer.gnss.configBlock[i].gnssId) {
                     _buffer.gnss.configBlock[i].resTrkCh = (_buffer.gnss.numTrkChHw - 3) /
(gnssCount * 2);
                     _buffer.gnss.configBlock[i].maxTrkCh = _buffer.gnss.numTrkChHw;
                  } else {
                     _buffer.gnss.configBlock[i].resTrkCh = 1;
                     _buffer.gnss.configBlock[i].maxTrkCh = 3;
                  }
                  _buffer.gnss.configBlock[i].flags = _buffer.gnss.configBlock[i].flags |
0x00000001;
               } else {
```

```
                _buffer.gnss.configBlock[i].resTrkCh = 0;
                _buffer.gnss.configBlock[i].maxTrkCh = 0;
                _buffer.gnss.configBlock[i].flags = _buffer.gnss.configBlock[i].flags &
0xFFFFFFFE;
            }
          }
          if (!memcmp(&start_gnss, &_buffer.gnss, sizeof(start_gnss))) {
            _send_message(CLASS_CFG, MSG_CFG_GNSS, &_buffer.gnss, 4 + (8 *
_buffer.gnss.numConfigBlocks));
            _unconfigured_messages |= CONFIG_GNSS;
            _cfg_needs_save = true;
          } else {
            _unconfigured_messages &= ~CONFIG_GNSS;
          }
        } else {
          _unconfigured_messages &= ~CONFIG_GNSS;
        }
        return false;
#endif

      case MSG_CFG_SBAS:
        if (gps._sbas_mode != 2) {
            Debug("Got SBAS settings %u %u %u 0x%x 0x%x\n",
              (unsigned)_buffer.sbas.mode,
              (unsigned)_buffer.sbas.usage,
              (unsigned)_buffer.sbas.maxSBAS,
              (unsigned)_buffer.sbas.scanmode2,
              (unsigned)_buffer.sbas.scanmode1);
          if (_buffer.sbas.mode != gps._sbas_mode) {
            _buffer.sbas.mode = gps._sbas_mode;
            _send_message(CLASS_CFG, MSG_CFG_SBAS,
&_buffer.sbas,
                          sizeof(_buffer.sbas));
            _unconfigured_messages |= CONFIG_SBAS;
            _cfg_needs_save = true;
          } else {
            _unconfigured_messages &= ~CONFIG_SBAS;
          }
        } else {
            _unconfigured_messages &= ~CONFIG_SBAS;
        }
        return false;
      case MSG_CFG_MSG:
        if(_payload_length == sizeof(ubx_cfg_msg_rate_6)) {
          // can't verify the setting without knowing the port
          // request the port again
          if(_ublox_port >= UBLOX_MAX_PORTS) {
            _request_port();
            return false;
```

```
            }
            _verify_rate(_buffer.msg_rate_6.msg_class, _buffer.msg_rate_6.msg_id,
                         _buffer.msg_rate_6.rates[_ublox_port]);
         } else {
            _verify_rate(_buffer.msg_rate.msg_class, _buffer.msg_rate.msg_id,
                         _buffer.msg_rate.rate);
         }
         return false;
      case MSG_CFG_PRT:
        _ublox_port = _buffer.prt.portID;
        return false;
      case MSG_CFG_RATE:
         if(_buffer.nav_rate.measure_rate_ms != gps._rate_ms[state.instance] ||
            _buffer.nav_rate.nav_rate != 1 ||
            _buffer.nav_rate.timeref != 0) {
            _configure_rate();
            _unconfigured_messages |= CONFIG_RATE_NAV;
            _cfg_needs_save = true;
         } else {
            _unconfigured_messages &= ~CONFIG_RATE_NAV;
         }
         return false;
      }

   }

   if (_class == CLASS_MON) {
      switch(_msg_id) {
      case MSG_MON_HW:
         if (_payload_length == 60 || _payload_length == 68) {
            log_mon_hw();
         }
         break;
      case MSG_MON_HW2:
         if (_payload_length == 28) {
            log_mon_hw2();
         }
         break;
      case MSG_MON_VER:
         _have_version = true;
         strncpy(_version.hwVersion, _buffer.mon_ver.hwVersion,
sizeof(_version.hwVersion));
         strncpy(_version.swVersion, _buffer.mon_ver.swVersion,
sizeof(_version.swVersion));
         GCS_MAVLINK::send_statustext_all(MAV_SEVERITY_INFO,
                                          "u-blox %d HW: %s SW: %s",
                                          state.instance + 1,
                                          _version.hwVersion,
                                          _version.swVersion);
```

```
            break;
        default:
            unexpected_message();
        }
        return false;
    }

#if UBLOX_RXM_RAW_LOGGING
    if (_class == CLASS_RXM && _msg_id == MSG_RXM_RAW && gps._raw_data != 0)
{
        log_rxm_raw(_buffer.rxm_raw);
        return false;
    } else if (_class == CLASS_RXM && _msg_id == MSG_RXM_RAWX &&
gps._raw_data != 0) {
        log_rxm_rawx(_buffer.rxm_rawx);
        return false;
    }
#endif // UBLOX_RXM_RAW_LOGGING

    if (_class != CLASS_NAV) {
        unexpected_message();
        return false;
    }

    switch (_msg_id) {
    case MSG_POSLLH:
        Debug("MSG_POSLLH next_fix=%u", next_fix);
        if (havePvtMsg) {
            _unconfigured_messages |= CONFIG_RATE_POSLLH;
            break;
        }
        _last_pos_time        = _buffer.posllh.time;
        state.location.lng    = _buffer.posllh.longitude;
        state.location.lat    = _buffer.posllh.latitude;
        state.location.alt    = _buffer.posllh.altitude_msl / 10;
        state.status          = next_fix;
        _new_position = true;
        state.horizontal_accuracy = _buffer.posllh.horizontal_accuracy*1.0e-3f;
        state.vertical_accuracy = _buffer.posllh.vertical_accuracy*1.0e-3f;
        state.have_horizontal_accuracy = true;
        state.have_vertical_accuracy = true;
#if UBLOX_FAKE_3DLOCK
        state.location.lng = 1491652300L;
        state.location.lat = -353632610L;
        state.location.alt = 58400;
        state.vertical_accuracy = 0;
        state.horizontal_accuracy = 0;
#endif
        break;
```

```
    case MSG_STATUS:
        Debug("MSG_STATUS fix_status=%u fix_type=%u",
              _buffer.status.fix_status,
              _buffer.status.fix_type);
        if (havePvtMsg) {
            _unconfigured_messages |= CONFIG_RATE_STATUS;
            break;
        }
        if (_buffer.status.fix_status & NAV_STATUS_FIX_VALID) {
            if( (_buffer.status.fix_type == AP_GPS_UBLOX::FIX_3D) &&
                (_buffer.status.fix_status & AP_GPS_UBLOX::NAV_STATUS_DGPS_USED)) {
                next_fix = AP_GPS::GPS_OK_FIX_3D_DGPS;
            }else if( _buffer.status.fix_type == AP_GPS_UBLOX::FIX_3D) {
                next_fix = AP_GPS::GPS_OK_FIX_3D;
            }else if (_buffer.status.fix_type == AP_GPS_UBLOX::FIX_2D) {
                next_fix = AP_GPS::GPS_OK_FIX_2D;
            }else{
                next_fix = AP_GPS::NO_FIX;
                state.status = AP_GPS::NO_FIX;
            }
        }else{
            next_fix = AP_GPS::NO_FIX;
            state.status = AP_GPS::NO_FIX;
        }
#if UBLOX_FAKE_3DLOCK
        state.status = AP_GPS::GPS_OK_FIX_3D;
        next_fix = state.status;
#endif
        break;
    case MSG_DOP:
        Debug("MSG_DOP");
        noReceivedHdop = false;
        state.hdop       = _buffer.dop.hDOP;
        state.vdop       = _buffer.dop.vDOP;
#if UBLOX_FAKE_3DLOCK
        state.hdop = 130;
        state.hdop = 170;
#endif
        break;
    case MSG_SOL:
        Debug("MSG_SOL fix_status=%u fix_type=%u",
              _buffer.solution.fix_status,
              _buffer.solution.fix_type);
        if (havePvtMsg) {
            state.time_week = _buffer.solution.week;
            break;
        }
        if (_buffer.solution.fix_status & NAV_STATUS_FIX_VALID) {
            if( (_buffer.solution.fix_type == AP_GPS_UBLOX::FIX_3D) &&
```

```
            (_buffer.solution.fix_status & AP_GPS_UBLOX::NAV_STATUS_DGPS_USED))
{
            next_fix = AP_GPS::GPS_OK_FIX_3D_DGPS;
         }else if( _buffer.solution.fix_type == AP_GPS_UBLOX::FIX_3D) {
            next_fix = AP_GPS::GPS_OK_FIX_3D;
         }else if (_buffer.solution.fix_type == AP_GPS_UBLOX::FIX_2D) {
            next_fix = AP_GPS::GPS_OK_FIX_2D;
         }else{
            next_fix = AP_GPS::NO_FIX;
            state.status = AP_GPS::NO_FIX;
         }
      }else{
         next_fix = AP_GPS::NO_FIX;
         state.status = AP_GPS::NO_FIX;
      }
      if(noReceivedHdop) {
         state.hdop = _buffer.solution.position_DOP;
      }
      state.num_sats    = _buffer.solution.satellites;
      if (next_fix >= AP_GPS::GPS_OK_FIX_2D) {
         state.last_gps_time_ms = AP_HAL::millis();
         state.time_week_ms    = _buffer.solution.time;
         state.time_week       = _buffer.solution.week;
      }
#if UBLOX_FAKE_3DLOCK
      next_fix = state.status;
      state.num_sats = 10;
      state.time_week = 1721;
      state.time_week_ms = AP_HAL::millis() + 3*60*60*1000 + 37000;
      state.last_gps_time_ms = AP_HAL::millis();
      state.hdop = 130;
#endif
      break;
   case MSG_PVT:
      Debug("MSG_PVT");
      havePvtMsg = true;
      // position
      _last_pos_time        = _buffer.pvt.itow;
      state.location.lng    = _buffer.pvt.lon;
      state.location.lat    = _buffer.pvt.lat;
      state.location.alt    = _buffer.pvt.h_msl / 10;
      switch (_buffer.pvt.fix_type)
      {
         case 0:
            state.status = AP_GPS::NO_FIX;
            break;
         case 1:
            state.status = AP_GPS::NO_FIX;
            break;
```

```
            case 2:
                state.status = AP_GPS::GPS_OK_FIX_2D;
                break;
            case 3:
                state.status = AP_GPS::GPS_OK_FIX_3D;
                if (_buffer.pvt.flags & 0b00000010)  // diffsoln
                    state.status = AP_GPS::GPS_OK_FIX_3D_DGPS;
                if (_buffer.pvt.flags & 0b01000000)  // carrsoln - float
                    state.status = AP_GPS::GPS_OK_FIX_3D_RTK_FLOAT;
                if (_buffer.pvt.flags & 0b10000000)  // carrsoln - fixed
                    state.status = AP_GPS::GPS_OK_FIX_3D_RTK_FIXED;
                break;
            case 4:
                GCS_MAVLINK::send_statustext_all(MAV_SEVERITY_INFO,
                        "Unexpected state %d", _buffer.pvt.flags);
                state.status = AP_GPS::GPS_OK_FIX_3D;
                break;
            case 5:
                state.status = AP_GPS::NO_FIX;
                break;
            default:
                state.status = AP_GPS::NO_FIX;
                break;
        }
        next_fix = state.status;
        _new_position = true;
        state.horizontal_accuracy = _buffer.pvt.h_acc*1.0e-3f;
        state.vertical_accuracy = _buffer.pvt.v_acc*1.0e-3f;
        state.have_horizontal_accuracy = true;
        state.have_vertical_accuracy = true;
        // SVs
        state.num_sats    = _buffer.pvt.num_sv;
        // velocity
        _last_vel_time         = _buffer.pvt.itow;
        state.ground_speed     = _buffer.pvt.gspeed*0.001f;          // m/s
        state.ground_course    = wrap_360(_buffer.pvt.head_mot * 1.0e-5f);       // Heading 2D
deg * 100000
        state.have_vertical_velocity = true;
        state.velocity.x = _buffer.pvt.velN * 0.001f;
        state.velocity.y = _buffer.pvt.velE * 0.001f;
        state.velocity.z = _buffer.pvt.velD * 0.001f;
        state.have_speed_accuracy = true;
        state.speed_accuracy = _buffer.pvt.s_acc*0.001f;
        _new_speed = true;
        // dop
        if(noReceivedHdop) {
            state.hdop       = _buffer.pvt.p_dop;
            state.vdop       = _buffer.pvt.p_dop;
        }
```

```
        state.last_gps_time_ms = AP_HAL::millis();

        // time
        state.time_week_ms    = _buffer.pvt.itow;
#if UBLOX_FAKE_3DLOCK
        state.location.lng = 1491652300L;
        state.location.lat = -353632610L;
        state.location.alt = 58400;
        state.vertical_accuracy = 0;
        state.horizontal_accuracy = 0;
        state.status = AP_GPS::GPS_OK_FIX_3D;
        state.num_sats = 10;
        state.time_week = 1721;
        state.time_week_ms = AP_HAL::millis() + 3*60*60*1000 + 37000;
        state.last_gps_time_ms = AP_HAL::millis();
        state.hdop = 130;
        next_fix = state.status;
#endif
        break;
    case MSG_VELNED:
        Debug("MSG_VELNED");
        if (havePvtMsg) {
            _unconfigured_messages |= CONFIG_RATE_VELNED;
            break;
        }
        _last_vel_time         = _buffer.velned.time;
        state.ground_speed     = _buffer.velned.speed_2d*0.01f;          // m/s
        state.ground_course    = wrap_360(_buffer.velned.heading_2d * 1.0e-5f);       // Heading
2D deg * 100000
        state.have_vertical_velocity = true;
        state.velocity.x = _buffer.velned.ned_north * 0.01f;
        state.velocity.y = _buffer.velned.ned_east * 0.01f;
        state.velocity.z = _buffer.velned.ned_down * 0.01f;
        state.ground_course = wrap_360(degrees(atan2f(state.velocity.y, state.velocity.x)));
        state.ground_speed = norm(state.velocity.y, state.velocity.x);
        state.have_speed_accuracy = true;
        state.speed_accuracy = _buffer.velned.speed_accuracy*0.01f;
#if UBLOX_FAKE_3DLOCK
        state.speed_accuracy = 0;
#endif
        _new_speed = true;
        break;
    case MSG_NAV_SVINFO:
        {
        Debug("MSG_NAV_SVINFO\n");
        static const uint8_t HardwareGenerationMask = 0x07;
        _hardware_generation = _buffer.svinfo_header.globalFlags &
HardwareGenerationMask;
```

```
        switch (_hardware_generation) {
            case UBLOX_5:
            case UBLOX_6:
                // only 7 and newer support CONFIG_GNSS
                _unconfigured_messages &= ~CONFIG_GNSS;
                break;
            case UBLOX_7:
            case UBLOX_M8:
#if UBLOX_SPEED_CHANGE
                port->begin(4000000U);
                Debug("Changed speed to 4Mhz for SPI-driven UBlox\n");
#endif
                break;
            default:
                hal.console->printf("Wrong Ublox Hardware Version%u\n",
_hardware_generation);
                break;
        };
        _unconfigured_messages &= ~CONFIG_VERSION;
        /* We don't need that anymore */
        _configure_message_rate(CLASS_NAV, MSG_NAV_SVINFO, 0);
        break;
        }
    default:
        Debug("Unexpected NAV message 0x%02x", (unsigned)_msg_id);
        if (++_disable_counter == 0) {
            Debug("Disabling NAV message 0x%02x", (unsigned)_msg_id);
            _configure_message_rate(CLASS_NAV, _msg_id, 0);
        }
        return false;
    }

    // we only return true when we get new position and speed data
    // this ensures we don't use stale data
    if (_new_position && _new_speed && _last_vel_time == _last_pos_time) {
        _new_speed = _new_position = false;
        return true;
    }
    return false;
}


// UBlox auto configuration

/*
 *  update checksum for a set of bytes
 */
void
```

```
AP_GPS_UBLOX::_update_checksum(uint8_t *data, uint16_t len, uint8_t &ck_a, uint8_t
&ck_b)
{
    while (len--) {
        ck_a += *data;
        ck_b += ck_a;
        data++;
    }
}


/*
 *  send a ublox message
 */
bool
AP_GPS_UBLOX::_send_message(uint8_t msg_class, uint8_t msg_id, void *msg, uint16_t
size)
{
    if (port->txspace() < (sizeof(struct ubx_header) + 2 + size)) {
        return false;
    }
    struct ubx_header header;
    uint8_t ck_a=0, ck_b=0;
    header.preamble1 = PREAMBLE1;
    header.preamble2 = PREAMBLE2;
    header.msg_class = msg_class;
    header.msg_id    = msg_id;
    header.length    = size;

    _update_checksum((uint8_t *)&header.msg_class, sizeof(header)-2, ck_a, ck_b);
    _update_checksum((uint8_t *)msg, size, ck_a, ck_b);

    port->write((const uint8_t *)&header, sizeof(header));
    port->write((const uint8_t *)msg, size);
    port->write((const uint8_t *)&ck_a, 1);
    port->write((const uint8_t *)&ck_b, 1);
    return true;
}

/*
 *  requests the given message rate for a specific message class
 *  and msg_id
 *  returns true if it sent the request, false if waiting on knowing the port
 */
bool
AP_GPS_UBLOX::_request_message_rate(uint8_t msg_class, uint8_t msg_id)
{
    // Without knowing what communication port is being used it isn't possible to verify
    // always ensure we have a port before sending the request
```

```
    if(_ublox_port >= UBLOX_MAX_PORTS) {
        _request_port();
        return false;
    } else {
        struct ubx_cfg_msg msg;
        msg.msg_class = msg_class;
        msg.msg_id    = msg_id;
        return _send_message(CLASS_CFG, MSG_CFG_MSG, &msg, sizeof(msg));
    }
}

/*
 *  configure a UBlox GPS for the given message rate for a specific
 *  message class and msg_id
 */
bool
AP_GPS_UBLOX::_configure_message_rate(uint8_t msg_class, uint8_t msg_id, uint8_t
rate)
{
    if (port->txspace() < (int16_t)(sizeof(struct ubx_header)+sizeof(struct
ubx_cfg_msg_rate)+2)) {
        return false;
    }

    struct ubx_cfg_msg_rate msg;
    msg.msg_class = msg_class;
    msg.msg_id    = msg_id;
    msg.rate      = rate;
    return _send_message(CLASS_CFG, MSG_CFG_MSG, &msg, sizeof(msg));
}

/*
 * save gps configurations to non-volatile memory sent until the call of
 * this message
 */
void
AP_GPS_UBLOX::_save_cfg()
{
    ubx_cfg_cfg save_cfg;
    save_cfg.clearMask = 0;
    save_cfg.saveMask = SAVE_CFG_ALL;
    save_cfg.loadMask = 0;
    _send_message(CLASS_CFG, MSG_CFG_CFG, &save_cfg, sizeof(save_cfg));
    _last_cfg_sent_time = AP_HAL::millis();
    _num_cfg_save_tries++;
    GCS_MAVLINK::send_statustext_all(MAV_SEVERITY_INFO,
                                     "GPS: u-blox %d saving config",
                                     state.instance + 1);
}
```

```
/*
  detect a Ublox GPS. Adds one byte, and returns true if the stream
  matches a UBlox
 */
bool
AP_GPS_UBLOX::_detect(struct UBLOX_detect_state &state, uint8_t data)
{
reset:
        switch (state.step) {
      case 1:
        if (PREAMBLE2 == data) {
            state.step++;
            break;
        }
        state.step = 0;
        /* no break */
      case 0:
        if (PREAMBLE1 == data)
            state.step++;
        break;
      case 2:
        state.step++;
        state.ck_b = state.ck_a = data;
        break;
      case 3:
        state.step++;
        state.ck_b += (state.ck_a += data);
        break;
      case 4:
        state.step++;
        state.ck_b += (state.ck_a += data);
        state.payload_length = data;
        break;
      case 5:
        state.step++;
        state.ck_b += (state.ck_a += data);
        state.payload_counter = 0;
        break;
      case 6:
        state.ck_b += (state.ck_a += data);
        if (++state.payload_counter == state.payload_length)
            state.step++;
        break;
      case 7:
        state.step++;
        if (state.ck_a != data) {
            state.step = 0;
                                goto reset;
```

```
        }
        break;
    case 8:
        state.step = 0;
                        if (state.ck_b == data) {
                                // a valid UBlox packet
                                return true;
                        } else {
                                goto reset;
                        }
    }
    return false;
}

void
AP_GPS_UBLOX::_request_version(void)
{
    _send_message(CLASS_MON, MSG_MON_VER, nullptr, 0);
}

void
AP_GPS_UBLOX::_configure_rate(void)
{
    struct ubx_cfg_nav_rate msg;
    // require a minimum measurement rate of 5Hz
    msg.measure_rate_ms = gps.get_rate_ms(state.instance);
    msg.nav_rate        = 1;
    msg.timeref         = 0;     // UTC time
    _send_message(CLASS_CFG, MSG_CFG_RATE, &msg, sizeof(msg));
}

static const char *reasons[] = {"navigation rate",
                        "posllh rate",
                        "status rate",
                        "solution rate",
                        "velned rate",
                        "dop rate",
                        "hw monitor rate",
                        "hw2 monitor rate",
                        "raw rate",
                        "version",
                        "navigation settings",
                        "GNSS settings",
                        "SBAS settings",
                        "PVT rate"};


void
AP_GPS_UBLOX::broadcast_configuration_failure_reason(void) const {
```

```
    for (uint8_t i = 0; i < ARRAY_SIZE(reasons); i++) {
        if (_unconfigured_messages & (1 << i)) {
            GCS_MAVLINK::send_statustext_all(MAV_SEVERITY_INFO, "GPS %d: u-blox
%s configuration 0x%02x",
                state.instance +1, reasons[i], _unconfigured_messages);
            break;
        }
    }
}

/*
  return velocity lag in seconds
 */
bool AP_GPS_UBLOX::get_lag(float &lag_sec) const
{
    switch (_hardware_generation) {
    case UBLOX_UNKNOWN_HARDWARE_GENERATION:
        lag_sec = 0.22f;
        // always bail out in this case, it's used to indicate we have yet to receive a valid
        // hardware generation, however the user may have inhibited us detecting the generation
        // so if we aren't allowed to do configuration, we will accept this as the default delay
        return gps._auto_config != AP_GPS::GPS_AUTO_CONFIG_ENABLE;
    case UBLOX_5:
    case UBLOX_6:
    default:
        lag_sec = 0.22f;
        break;
    case UBLOX_7:
    case UBLOX_M8:
        // based on flight logs the 7 and 8 series seem to produce about 120ms lag
        lag_sec = 0.12f;
        break;
    };
    return true;
}

void AP_GPS_UBLOX::Write_DataFlash_Log_Startup_messages() const
{
    AP_GPS_Backend::Write_DataFlash_Log_Startup_messages();

    if (_have_version) {
        DataFlash_Class::instance()->Log_Write_MessageF("u-blox %d HW: %s SW: %s",
                                        state.instance+1,
                                        _version.hwVersion,
                                        _version.swVersion);
    }
}
```

Library* APM 2.8 untuk Motor *Brushless

```
/*
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

/*
 *       AP_MotorsSingle.cpp - ArduCopter motors library
 *       Code by RandyMackay. DIYDrones.com
 *
 */

#include <AP_HAL/AP_HAL.h>
#include <AP_Math/AP_Math.h>
#include "AP_MotorsCoax.h"
#include <GCS_MAVLink/GCS.h>

extern const AP_HAL::HAL& hal;


// init
void AP_MotorsCoax::init(motor_frame_class frame_class, motor_frame_type frame_type)
{
    _servo1 = SRV_Channels::get_channel_for(SRV_Channel::k_motor1, CH_1);
    _servo2 = SRV_Channels::get_channel_for(SRV_Channel::k_motor2, CH_2);
    _servo3 = SRV_Channels::get_channel_for(SRV_Channel::k_motor3, CH_3);
    _servo4 = SRV_Channels::get_channel_for(SRV_Channel::k_motor4, CH_4);
    if (!_servo1 || !_servo2 || !_servo3 || !_servo4) {
        GCS_MAVLINK::send_statustext_all(MAV_SEVERITY_ERROR,          "MotorsCoax:
unable to setup output channels");
        // don't set initialised_ok
        return;
    }

    // set the motor_enabled flag so that the main ESC can be calibrated like other frame types
    motor_enabled[AP_MOTORS_MOT_5] = true;
    motor_enabled[AP_MOTORS_MOT_6] = true;

    // we set four servos to angle
```

```
    _servo1->set_angle(AP_MOTORS_COAX_SERVO_INPUT_RANGE);
    _servo2->set_angle(AP_MOTORS_COAX_SERVO_INPUT_RANGE);
    _servo3->set_angle(AP_MOTORS_COAX_SERVO_INPUT_RANGE);
    _servo4->set_angle(AP_MOTORS_COAX_SERVO_INPUT_RANGE);

    // record successful initialisation if what we setup was the desired frame_class
    _flags.initialised_ok = (frame_class == MOTOR_FRAME_COAX);
}

// set frame class (i.e. quad, hexa, heli) and type (i.e. x, plus)
void AP_MotorsCoax::set_frame_class_and_type(motor_frame_class frame_class,
motor_frame_type frame_type)
{
    _flags.initialised_ok = (frame_class == MOTOR_FRAME_COAX);
}

// set update rate to motors - a value in hertz
void AP_MotorsCoax::set_update_rate( uint16_t speed_hz )
{
    // record requested speed
    _speed_hz = speed_hz;

    uint32_t mask =
        1U << AP_MOTORS_MOT_5 |
        1U << AP_MOTORS_MOT_6 ;
    rc_set_freq(mask, _speed_hz);
}

// enable - starts allowing signals to be sent to motors
void AP_MotorsCoax::enable()
{
    // enable output channels
    rc_enable_ch(AP_MOTORS_MOT_1);
    rc_enable_ch(AP_MOTORS_MOT_2);
    rc_enable_ch(AP_MOTORS_MOT_3);
    rc_enable_ch(AP_MOTORS_MOT_4);
    rc_enable_ch(AP_MOTORS_MOT_5);
    rc_enable_ch(AP_MOTORS_MOT_6);
}

void AP_MotorsCoax::output_to_motors()
{
    switch (_spool_mode) {
        case SHUT_DOWN:
            // sends minimum values out to the motors
            rc_write(AP_MOTORS_MOT_1,     calc_pwm_output_1to1(_roll_radio_passthrough,
_servo1));
            rc_write(AP_MOTORS_MOT_2,   calc_pwm_output_1to1(_pitch_radio_passthrough,
_servo2));
```

```
            rc_write(AP_MOTORS_MOT_3,    calc_pwm_output_1to1(-_roll_radio_passthrough,
_servo3));
            rc_write(AP_MOTORS_MOT_4,  calc_pwm_output_1to1(-_pitch_radio_passthrough,
_servo4));
            rc_write(AP_MOTORS_MOT_5, get_pwm_output_min());
            rc_write(AP_MOTORS_MOT_6, get_pwm_output_min());
            break;
        case SPIN_WHEN_ARMED:
            // sends output to motors when armed but not flying
            rc_write(AP_MOTORS_MOT_1,         calc_pwm_output_1to1(_spin_up_ratio         *
_actuator_out[0], _servo1));
            rc_write(AP_MOTORS_MOT_2,         calc_pwm_output_1to1(_spin_up_ratio         *
_actuator_out[1], _servo2));
            rc_write(AP_MOTORS_MOT_3,         calc_pwm_output_1to1(_spin_up_ratio         *
_actuator_out[2], _servo3));
            rc_write(AP_MOTORS_MOT_4,         calc_pwm_output_1to1(_spin_up_ratio         *
_actuator_out[3], _servo4));
            rc_write(AP_MOTORS_MOT_5, calc_spin_up_to_pwm());
            rc_write(AP_MOTORS_MOT_6, calc_spin_up_to_pwm());
            break;
        case SPOOL_UP:
        case THROTTLE_UNLIMITED:
        case SPOOL_DOWN:
            // set motor output based on thrust requests
            rc_write(AP_MOTORS_MOT_1, calc_pwm_output_1to1(_actuator_out[0], _servo1));
            rc_write(AP_MOTORS_MOT_2, calc_pwm_output_1to1(_actuator_out[1], _servo2));
            rc_write(AP_MOTORS_MOT_3, calc_pwm_output_1to1(_actuator_out[2], _servo3));
            rc_write(AP_MOTORS_MOT_4, calc_pwm_output_1to1(_actuator_out[3], _servo4));
            rc_write(AP_MOTORS_MOT_5, calc_thrust_to_pwm(_thrust_yt_ccw));
            rc_write(AP_MOTORS_MOT_6, calc_thrust_to_pwm(_thrust_yt_cw));
            break;
    }
}
// get_motor_mask - returns a bitmask of which outputs are being used for motors or servos (1
means being used)
//  this can be used to ensure other pwm outputs (i.e. for servos) do not conflict
uint16_t AP_MotorsCoax::get_motor_mask()
{
    uint32_t mask =
        1U << AP_MOTORS_MOT_1 |
        1U << AP_MOTORS_MOT_2 |
        1U << AP_MOTORS_MOT_3 |
        1U << AP_MOTORS_MOT_4 |
        1U << AP_MOTORS_MOT_5 |
        1U << AP_MOTORS_MOT_6;
    return rc_map_mask(mask);
}

// sends commands to the motors
```

```
void AP_MotorsCoax::output_armed_stabilizing()
{
    float   roll_thrust;                // roll thrust input value, +/- 1.0
    float   pitch_thrust;               // pitch thrust input value, +/- 1.0
    float   yaw_thrust;                 // yaw thrust input value, +/- 1.0
    float   throttle_thrust;            // throttle thrust input value, 0.0 - 1.0
    float   thrust_min_rpy;             // the minimum throttle setting that will not limit the roll and pitch output
    float    thr_adj;                   // the difference between the pilot's desired throttle and throttle_thrust_best_rpy
    float   thrust_out;                 //
    float   rp_scale = 1.0f;            // this is used to scale the roll, pitch and yaw to fit within the motor limits
    float   actuator_allowed = 0.0f;    // amount of yaw we can fit in

    // apply voltage and air pressure compensation
    roll_thrust = _roll_in * get_compensation_gain();
    pitch_thrust = _pitch_in * get_compensation_gain();
    yaw_thrust = _yaw_in * get_compensation_gain();
    throttle_thrust = get_throttle() * get_compensation_gain();

    // sanity check throttle is above zero and below current limited throttle
    if (throttle_thrust <= 0.0f) {
        throttle_thrust = 0.0f;
        limit.throttle_lower = true;
    }
    if (throttle_thrust >= _throttle_thrust_max) {
        throttle_thrust = _throttle_thrust_max;
        limit.throttle_upper = true;
    }
    _throttle_avg_max = constrain_float(_throttle_avg_max, throttle_thrust, _throttle_thrust_max);

    float rp_thrust_max = MAX(fabsf(roll_thrust), fabsf(pitch_thrust));

    // calculate how much roll and pitch must be scaled to leave enough range for the minimum yaw
    if (is_zero(rp_thrust_max)) {
        rp_scale = 1.0f;
    } else {
        rp_scale = constrain_float((1.0f - MIN(fabsf(yaw_thrust), 0.5f*(float)_yaw_headroom/1000.0f)) / rp_thrust_max, 0.0f, 1.0f);
        if (rp_scale < 1.0f) {
            limit.roll_pitch = true;
        }
    }

    actuator_allowed = 2.0f * (1.0f - rp_scale * rp_thrust_max);
    if (fabsf(yaw_thrust) > actuator_allowed) {
```

```
        yaw_thrust = constrain_float(yaw_thrust, -actuator_allowed, actuator_allowed);
        limit.yaw = true;
    }

    // calculate the minimum thrust that doesn't limit the roll, pitch and yaw forces
    thrust_min_rpy = MAX(fabsf(rp_scale * rp_thrust_max), fabsf(yaw_thrust));

    thr_adj = throttle_thrust - _throttle_avg_max;
    if (thr_adj < (thrust_min_rpy - _throttle_avg_max)) {
        // Throttle can't be reduced to the desired level because this would mean roll or pitch control
        // would not be able to reach the desired level because of lack of thrust.
        thr_adj = MIN(thrust_min_rpy, _throttle_avg_max) - _throttle_avg_max;
    }

    // calculate the throttle setting for the lift fan
    thrust_out = _throttle_avg_max + thr_adj;

    if (fabsf(yaw_thrust) > thrust_out) {
        yaw_thrust = constrain_float(yaw_thrust, -thrust_out, thrust_out);
        limit.yaw = true;
    }

    _thrust_yt_ccw = thrust_out + 0.5f * yaw_thrust;
    _thrust_yt_cw = thrust_out - 0.5f * yaw_thrust;

    // limit thrust out for calculation of actuator gains
    float thrust_out_actuator = constrain_float(MAX(_throttle_hover*0.5f,thrust_out), 0.1f, 1.0f);
    if (is_zero(thrust_out)) {
        limit.roll_pitch = true;
    }
    // force of a lifting surface is approximately equal to the angle of attack times the airflow velocity squared
    // static thrust is proportional to the airflow velocity squared
    // therefore the torque of the roll and pitch actuators should be approximately proportional to
    // the angle of attack multiplied by the static thrust.
    _actuator_out[0] = roll_thrust/thrust_out_actuator;
    _actuator_out[1] = pitch_thrust/thrust_out_actuator;
    if (fabsf(_actuator_out[0]) > 1.0f) {
        limit.roll_pitch = true;
        _actuator_out[0] = constrain_float(_actuator_out[0], -1.0f, 1.0f);
    }
    if (fabsf(_actuator_out[1]) > 1.0f) {
        limit.roll_pitch = true;
        _actuator_out[1] = constrain_float(_actuator_out[1], -1.0f, 1.0f);
    }
    _actuator_out[2] = -_actuator_out[0];
    _actuator_out[3] = -_actuator_out[1];
```

```
}
// output_test - spin a motor at the pwm value specified
//  motor_seq is the motor's sequence number from 1 to the number of motors on the frame
//  pwm value is an actual pwm value that will be output, normally in the range of 1000 ~ 2000
void AP_MotorsCoax::output_test(uint8_t motor_seq, int16_t pwm)
{
    // exit immediately if not armed
    if (!armed()) {
        return;
    }
    // output to motors and servos
    switch (motor_seq) {
        case 1:
            // flap servo 1
            rc_write(AP_MOTORS_MOT_1, pwm);
            break;
        case 2:
            // flap servo 2
            rc_write(AP_MOTORS_MOT_2, pwm);
            break;
        case 3:
            // flap servo 3
            rc_write(AP_MOTORS_MOT_3, pwm);
            break;
        case 4:
            // flap servo 4
            rc_write(AP_MOTORS_MOT_4, pwm);
            break;
        case 5:
            // motor 1
            rc_write(AP_MOTORS_MOT_5, pwm);
            break;
        case 6:
            // motor 2
            rc_write(AP_MOTORS_MOT_6, pwm);
            break;
        default:
            // do nothing
            break;
    }
}
```

LAMPIRAN C

Biaya Pembelian Alat dan Bahan untuk Pembuatan USV

Biaya Pembelian Alat dan Bahan Purwarupa USV

No	Barang	Jumlah	Harga Satuan	Harga	Keterangan
1	Karton	5 lembar	Rp9,300.00	Rp12,700.00	Beli
2	Lem G	11 buah	Rp7,500.00	Rp82,500.00	Beli
3	Resin	1.5 kg	Rp17,000.00	Rp51,000.00	Beli
4	Katalis	1 buah		Rp9,000.00	Beli
5	Kuas	6 buah	Rp3,500.00	Rp21,000.00	Beli
6	Mat Fiber	1 kg		Rp33,000.00	Beli
7	*Pylox Surfacer*	2 buah	Rp23,000.00	Rp46,000.00	Beli
8	Dempul Alfagloss	1 kg		Rp32,500.00	Beli
9	Amplas 80	0.5 m	Rp8,000.00	Rp4,000.00	Beli
10	Amplas 400	2 lembar	Rp4,500.00	Rp9,000.00	Beli
11	Amplas 1000	2 lembar	Rp5,000.00	Rp10,000.00	Beli
12	Lem Dextone	2 set	Rp14,000.00	Rp28,000.00	Beli
13	PVC Foam 3 mm	1 lembar		Rp60,000.00	Beli
14	PVC Foam 5 mm	2 lembar	Rp75,000.00	Rp150,000.00	Beli
15	Solasi Kertas	1 buah		Rp3,000.00	Beli
16	Scotlight	0.5 m	Rp33,000.00	Rp16,500.00	Beli
17	Double Tape 3M	2 buah	Rp10,000.00	Rp10,000.00	Beli
18	Baut M3	10 buah	Rp300.00	Rp3,000.00	Beli
19	*Cable Ties*	6 buah	Rp2,500.00	Rp15,000.00	Beli
20	*Rudder*	2 buah	Rp130,000.00	Rp260,000.00	Beli
20	Ruji	4 buah	Rp1,000.00	Rp4,000.00	Beli
21	*Shaft Tunnel & Shaft*	2 buah	Rp200,000.00	Rp400,000.00	Pinjam
22	*Propeller*	2 buah	Rp40,000.00	Rp80,000.00	Pinjam
23	Motor *Brushless*	2 buah	Rp500,000.00	Rp100,000.00	Pinjam
24	*Rubber*	2 buah	Rp50,000.00	Rp100,000.00	
24	*Joint*	2 buah	Rp150,000.00	Rp300,000.00	Pinjam
25	*Electronic Speed Control*	2 buah	Rp450,000.00	Rp900,000.00	Pinjam
26	Motor Servo Tower Pro MG 945	2 buah	Rp80,000.00	Rp160,000.00	Beli
27	Regulator	1 buah	Rp12,000.00	Rp12,000.00	Beli
28	ArduPilot Mega 2.8	1 buah		Rp800,000.00	Pinjam
29	Baterai Li-Po	3 buah	Rp250,000.00	Rp750,000.00	Beli
30	*Radio Control & Receiver*	1 set		Rp650,000.00	Pinjam

31	*Power Bank*	1 buah		Rp300,000.00	Pinjam
32	*Radio Telemetry*	1 set		Rp270,000.00	Pinjam
33	*GoPro Hero 4*	1 buah		Rp5,200,000.00	Pinjam
34	*Image Transmitter*	1 set		Rp50,000.00	Pinjam
35	Kabel	1 set		Rp200,000.00	Beli
	TOTAL			**Rp11,132,200.00**	

Pembelian Alat dan Bahan Tugas Akhir USV

No	Tanggal/Toko	Barang	Jumlah	Harga Satuan	Harga Total
1	19 April 2017/Mitra Media	Lem G	2	Rp 6.350	Rp25,500.00
		Karton	5	Rp 9.300	Rp12,700.00
2	21 April 2017/Victory	Resin	1.5 kg	Rp 33.000	Rp44,000.00
		Katalis	1		Rp9,000.00
		Kuas	2	Rp 3.500	Rp7,000.00
		Mat Fiber	1 kg		Rp33,000.00
3	22 April 2017/Victory	Resin	0.5 kg		Rp15,000.00
		Kuas	1		Rp3,500.00
4	22 April 2017/UD. Mandiri	*Pylox Surfacer*	1		Rp22,000.00
		Dempul Alfagloss	1 kg		Rp32,500.00
5	25 April 2017/Victory	Resin	0.5 kg		Rp17,000.00
		Kuas	1		Rp3,500.00
6	26 April 2017/Victory	Resin	0.5 kg		Rp17,000.00
		Kuas	2		Rp7,000.00
7	03 Mei 2017/Merdeka	PVC Foam 3 mm	1		Rp60,000.00
		PVC Foam 5 mm	2	Rp 75.000	Rp150,000.00
8	03 Mei 2017/UD. Mandiri	Amplas 400	2	Rp 4.500	Rp9,000.00
		Amplas 80	0.5 m	Rp 8.000	Rp4,000.00
9	17 Mei 2017/Mitra Media	Lem G	4	Rp 6.375	Rp25,500.00
10	27 Mei 2017/Ojo Lali	Lem Dextone	1		Rp14,000.00
11	27 Mei 2017/ I See	Baterai LiPo 2200	1		Rp250,000.00
		Arduino Uno	1		Rp80,000.00
12	29 Mei 2017/Ojo Lali	Lem Dextone	1		Rp14,000.00
13	31 Mei 2017/UD. Mandiri	*Pylox Surfacer*	1		Rp23,000.00
14	01 Juni 2017/Akhi Shop	STM32F4	1		Rp370,000.00
		Go Jek			Rp15,000.00
15	02 Juni 2017/Tokopedia	Regulator	4	Rp 12.000	Rp48,000.00
		Go Jek			Rp15,000.00
16	03 Juni 2017/Tokopedia	Motor Servo Tower Pro MG 945	2	Rp 80.000	Rp160,000.00
		Go Jek			Rp15,000.00
17	04 Juni 2017/Tokopedia	*Rudder*	2	Rp130.000	Rp260,000.00
		Go Jek			Rp15,000.00
18	05 Juni 2017/Mitra Media	Lem G	5	Rp 7.400	Rp37,000.00

19	05 Juni 2017/UD. Mandiri	Amplas 400	2	Rp 4.500	Rp9,000.00
		Pylox	2	Rp 23.000	Rp46,000.00
20	06 Juni 2017/Ojo Lali	*Cable Ties*	6	Rp 2.500	Rp15,000.00
		Ruji	4	Rp 1.000	Rp4,000.00
21	08 Juni 2017/Ojo Lali	Baut M3	10	Rp 300	Rp3,000.00
		Pylox	1		Rp22,000.00
22	10 Juni 2017/Ojo Lali	Solasi Kertas	1		Rp3,000.00
		Double Tape 3M	2	Rp 10000	Rp20,000.00
23	10 Juli 2017/Papyrus	Scotlight	0.5 m	Rp 33000	Rp16,500.00
				TOTAL	**Rp1,951,700.00**

BIODATA PENULIS

ERICZA DAMARANDA S dilahirkan di Malang, 1Juni 1995. Penulis merupakan anak pertama dari 3 bersaudara dalam keluarga. Dibesarkan di Malang dan menamatkan pendidikan formal tingkat MI di MI Jenderal Sudirman Malang, tingkat SMP di SMPN 20 Malang dan tingkat MA di MAN 3 Malang hingga melanjutkan pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya. Penulis diterima di Jurusan Teknik Perkapalan, Fakultas Teknologi Kelautan ITS pada tahun 2013 melalui jalur SBMPTN.

Di Jurusan Teknik Perkapalan, Penulis mengambil Bidang Studi Rekayasa Perkapalan – Desain Kapal. Selama masa studi di ITS, Penulis aktif berkegiatan di Himpunan Mahasiswa Teknik Perkapalan (HIMATEKPAL) sebagai staf biro Hidromodeling 2014-2015, dan menjadi staf ahli biro Hidromodeling 2014-2016. Selain itu, Penulis juga aktif berkegiatan di Unit Kegiatan Mahasiswa (UKM) yaitu UKM Maritime Challenge sebagai staf divisi Sosial dan Budaya 2014-2015, dan UKM Robotika ITS sebagai koordinator mekanik divisi Roboboat 2015-2016. Penulis juga sempat mengikuti beberapa pelatihan , baik pelatihan pembentukan *soft skill* seperti LKMM dan pelatihan yang menunjang kebutuhan akademis selama perkuliahan, seperti pelatihan perangkat lunak AutoCAD dan Maxsurf. Tidak ada yang spesial dari segi prestasi penulis selama masa perkuliahan, kecuali menjuarai lomba Nasional sebanyak 4 kali (di ITS, UI, dan UNDIP) dan lomba Internasional sebanyak 2 kali (di Amerika Serikat dan Singapura).

Email: ericza45@gmail.com / ericza.na13@gmail.com
Phone: +62 822 342 812 73